# A Golf Scoring Information System

Michael Walters

Computing

2007

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)_____

# Summary

This project aims to deliver an information system, which provides all the necessary functionality involved with calculating results for golf competitions. It is intended that this system will not only be competitive with existing commercial solutions but will also provide a niche with some additional functionality creating a competitive advantage. The software is intended for use by golf club secretaries and professionals, so that they are able to manage both their members and competitions effectively.

Before implementing a solution, much background research and analysis was conducted and this knowledge was then used when producing a system design. The implemented solution was then fully tested and evaluated. Mr.Tulley at Monmouth Golf Club as well as some general users have evaluated the complete working solution. The results from these can be found in the evaluation section.

## Acknowledgments

I would like to thank Haiko Muller for his help throughout my project. All his advice from the initial background research up to the evaluation phase of system development has been utilized when facing obstacles throughout this process.

In addition to this, many thanks go to Mr.Tulley at Monmouth Golf Club for his time and comments during the presentation. Finally I would like to thank Ken Brodlie for his advice on how I could further this system.

# Contents

**Chapter 1 – Analysis**

1.1 Aim

1.2 Objectives

1.3 Problem Domain

1.3.1 Club 2000

1.3.2 Handicap Master

1.3.3 Jencess Connect Suite

1.3.4 Golf Computer Systems Membership Management

1.3.5 Conclusions

1.4 Proposed Solution

1.5 Project Requirements

1.5.1 Minimum Requirements

1.5.2 Advanced Requirements

1.5.3 Extensions On Advanced Requirements

1.6 Relevance To Degree

1.7 Project Structure

**Chapter 2 – Background Research**

2.1 Introduction

2.2 Usability

2.2.1 Nielsen's Usability Principles

2.2.2 The Principles Of Interface Design

2.3 System Methodologies

2.3.1 The Waterfall Approach

2.3.2 Iterative Development

2.3.3 Component Based Development

- Throwaway Prototyping

- Evolutionary Prototyping

- Incremental Prototyping

2.3.4 Methodology Conclusions

2.4 Database Technologies

2.4.1 Fundamental Requirements Of The Database

2.4.2 Database Management Systems

- Postgres

- Oracle

- Microsoft SQL Server

- MySQL

2.4.3 DBMS Conclusions

- Postgres and MySQL

- Postgres and Oracle

- Postgres and Microsoft SQL Server

2.5 Programming Languages

2.5.1 JAVA

2.5.2 C++

2.5.3 Programming Language Conclusions

2.6 Development Tools

2.6.1 Eclipse

2.6.2 IntelliJ IDEA

2.6.3 Development Tool Conclusions

## Chapter 3 – System Design

3.1 Introduction

3.2 System Architecture

3.3 Incorporating System Functionality Into Design

3.4 Competition Process

3.5 Database Design

3.5.1 Search Queries

3.6 User Interfaces

3.6.1 Colouring

3.6.2 Navigation

3.6.3 Graphics

**Chapter 4 – Development Of The Solution**

4.1 Introduction

4.2 Component: Database

4.2.1 First Iteration

4.2.2 Second Iteration

4.3 Component: User Interface

4.3.1 First Iteration

- Development Of Main Interface

- Development Of Add Member Interface

4.4 Component: Back End Classes

4.4.1 First Iteration

4.4.2 Second Iteration

4.4.2 Third Iteration

4.5 Component: Star Members Advanced Requirement

**Chapter 5 – Testing**

5.1 Introduction

5.2 Compatibility Testing

5.2.1 Windows XP Home 5.1.2600

5.2.2 Fedora Core 6

5.3 Basic System Testing

5.3.1 Result Calculation Testing

- Hole Data

- Gross Score Calculation Test

- Net Score Calculation Test

- Stableford Score Calculation Test

- Conclusions From Testing Competition Results

5.3.2 Login Testing

5.3.3 Competition Constraint Testing

5.3.4 Validation Testing

## Chapter 6 – Evaluation

6.1 Introduction

6.2 Evaluation Against Minimum Requirements

6.2.1 A Database Containing Golf Members Information

6.2.2 The System Must Allow Data Entry From Scorecards

6.2.3 Output Must Include A List Of Competition Results After Data Processing

6.2.4 Must Have The Ability To Deal With Changes To The Database Through A UI

6.3 Evaluation By Monmouth Golf Club

6.3.1 Preparing For The Presentation

6.3.2 The Presentation

6.3.3 Presentation Feedback

6.4 Feedback From Users

6.4.1 General Users

6.5 Future Project Extensions

6.5.1 SSS Inclusion

6.5.2 Members Best Hole Facility

6.5.3 Lesson Booking System

6.6 Evaluation Of Project Management

6.6.1 Did I Manage To Stick To The Schedule?

6.6.2 Was The Schedule Realistic?

***Bibliography***

***Appendix A – My Personal Reflections***

***Appendix B – Project Schedule***

***Appendix C – Glossary Of Golf Terminology***

***Appendix D – Presentation Material***

***Appendix E – Database Tables***

***Appendix F – User Interface Work flow***

***Appendix G – Stableford Points Calculations***

***Appendix H – Use Case Diagram for Registration Process***

***Appendix I – Work flow Diagram of Competition Calculations***

***Appendix J – Nielsen's Usability Principles***

***Appendix K – Proposed Graphical User Interface Layout***

***Appendix L – System Requirements***

***User Manual***

# Chapter 1 – Project Overview

## 1.1 Aim

This project aims to design and implement an information system, which provides the functionality of a golf scoring system. The main objectives are to streamline competition entry, automatically calculate a list of competition results after data input and allow for member management. The product is aimed at a club secretary and professional who together run golf clubs as a whole. It is these personnel who will have the opportunity to add, edit or remove members or competitions. Additionally, they will also be required to input raw competition data into this information system so that they can obtain the results from the application.

There is also scope to incorporate both financial and schedule information into the system to create a more complete golfing application. As far as these are concerned, the financial aspect would involve checking member's annual fees and alerting certain people if they haven't paid. Also, schedule information involving both an annual calendar in order to keep track of social events in the clubhouse and also a calendar intended for the club professional to book lessons with members.

## 1.2 Objectives

If the project aims are to be satisfied, there are a few objectives of importance, which need to be fulfilled. To begin, the developer will undertake thorough investigation of the current systems in place so that the analysis and evaluation of these will allow for a clear understanding of the problem situation. In addition to this, methodologies must be researched so that an appropriate schedule can be produced which is desired to run in tandem with this object-oriented approach to architecture. It is imperative that the users needs are both identified and met and in doing this it is essential to find the correct balance between resources, deadlines and features in this trade off triangle. If done correctly any potential bottlenecks that may lead to delay or wasted resources should be avoided.

A further objective of this golf scoring system is that the graphical user interface must achieve consistency whilst enhancing the usability of the underlying logical design. In achieving this it must abide by any relevant human-computer interaction principles. Once the creation of the system is complete, an essential objective for the systems analyst is thorough testing to ensure any errors have been removed. Evaluation must then take place with regards to establishing whether all the initial user requirements of the system have been fulfilled.

## 1.3 The Problem Domain

In order to fully determine the extent of the problem it was in the interests of the developer to investigate existing systems in several golf clubs. Mr. Peter Tulley the current secretary at Monmouth Golf Club in Gwent, South Wales was first contacted in order to discover the current handicap and membership systems utilized by the club. Mr. Tulley stated that the systems provided by Club 2000 are currently in use by the golf club to maintain members details and provide a tool for the handicapping system. Having paid a visit to this Golf Club it was clear that the system fulfilled much of the necessary functionality that would be expected of such a software solution. This includes managing members details, competition data and producing score sheet results from competitions.

Having contacted several other golf club secretaries it has become apparent that many existing golf-scoring systems being used are of limited capability. This is because many systems only provide basic functionality to perform the necessary golf scoring calculations. Therefore, there is scope for a system which not only provides this level of service but also contains some additional interesting features. For example, this could be the inclusion of a star members feature, which displays the records set by members, who have participated in weekly competitions. In order to discover the rules of golf (some of which will be utilized by the proposed system) visit [7].

### 1.3.1 Club 2000

There are some systems in place, which can function to a high level of capability. For example Club 2000 provide a membership administration system which has the capabilities to maintain up to 30,000 members records in an unlimited number of categories. Furthermore they also provide a handicapping system, which is broken down into a number of core areas, within each area the systems can adapt to almost any clubs requirement, as the system is highly flexible. Additionally Club 2000 also provide PSI (Player Score Input) systems where players can electronically transmit their scores rather than using score cards and posting them into a box at the end of a players round. Although this has the advantage of the scores being downloaded into a handicap system so no input is needed, it removes long lasting traditions of the game. This is because at all levels of golf, score cards are still used for competitions as it is part of the history of the sport and therefore this idea has not been a success. Further information on Club 2000 can be found from [2].

### 1.3.2 Handicap Master

Another commercial system is HandicapMaster golf software. HandicapMaster is a program for running Golf Competitions and managing Handicaps. All editions of HandicapMaster come with a comprehensive list of features. These include:

- Process Stroke play, Stableford, Par, Foursomes, Greensomes and Four Ball Better Ball Competitions.
- Process Multiple-round or Alternative-day competitions.
- Run Match-play Knockout Draws.
- Membership Records.
- Free uploading of Handicaps and Competition Results.
- Automatic software updates.

For more information visit [9].

### 1.3.3 Jencess Connect Suite

Jencess Connect Suite provide software solutions covering many aspects of a complete golfing system. They release these in sections like finance, handicapping, reservations and point of sale. They provide a solution to the problem of controlling the handicapping system using score-way handicapping. This allows members to enter their scores with touch screen, mouse or keyboard and calculates their updated handicap. It also provides a member service center which allows members to book and view their own tee-times, enter club tournaments and enter ballots. Score-way also brings an element of security allowing members to view but not change another member's information and club ranking. Club management can set security privileges for each staff member in addition to this. Detailed member information is also stored and maintained including whether they have an active or inactive status. Also stored are their locker number, membership type and rounds played. Using this they can easily add, edit or delete names, addresses, phone numbers and member codes. Another impressive feature of the Score-way handicapping system is that it provides on line access aided by ClubAssistant.net. This service allows members to input scores through the Internet as if they were at the club. In addition to this members are able to view handicap information and their last 20 scores all in real time. For more information visit the URL at [12].

As previously mentioned, the Jencess Connect Suite offers many software product solutions for a complete golf system. Therefore, in addition to the system described above they also offer Acct-way, which is a financial management system. This is of importance because organizing club finances is part of a clubs main functionalities. This system is relatively easy to setup using a clear, intuitive interface it allows users to quickly manage general ledger accounts, receivables, payables, inventory and beginning balances. The idea behind Acct-way is that users are able to enter all of their employees, vendors and members and have the flow through to all modules. As far as membership is concerned the main functionalities are designed for accounts being receivable, member billing, food and beverage minimums as well as invoicing and payments. This financial management system tracks budget and actual financial data in order to aid the user in tracking their revenues and expenses, maximizing profits and managing their overall company.

### 1.3.4 Golf Computer Systems Membership Management

A final system, which also requires discussion, is that provided by Golf Computer Systems Membership Management. This system is a comprehensive and highly flexible membership management software application, which incorporates recording, reporting, billing and marketing functions. The system is highly configurable to enable its presentation and behavior to be tailored to suit the needs of different types of membership situations. The system shares data with other applications from Golf Computer Systems such as bookings, POS and golf management, to ensure that entry of information does not need duplicating. More information can be in [5].

### 1.3.5 Conclusions

Although the systems discussed provide much intended functionality that this proposed golf scoring information system will have, they have one noticeable disadvantage. These competitive systems are extremely costly. They also appear to cover sections of a complete golf system, for example with the software solutions that Jencess provide as these are split into handicapping, management and finance systems. Therefore there is a need for a complete system, which provides all the required functionality in order to overcome problems occurring everyday at a golf club. This provides users with a tool, which can meet the criteria of any task which needs completing at a golf club, whether this maybe checking membership accounts, adding members to the club or inserting competition data. It is desired that the proposed system will overcome this problem and it will provide some additional features which further enhance it's capabilities. For example, an eclectic scoring mechanism could be created which discovers each members lowest overall score.

After thorough investigation of these existing systems it has become clear that there is a hole in the market for a system, which is cheap in cost, and which provides all the necessary functionality a golf scoring system requires. This is because the current systems being utilized are either far to basic or they seem far to expensive for this type of application and therefore this object oriented approach to architecture will be designed with the intention of lowering costs whilst still overcoming all of the necessary requirements.

## 1.4 Proposed Solution

A system, which stores and maintains golf club information and also provides functionality for the production of weekly competition results, is the desired solution. The key idea behind such an application is that it gives the users (golf club secretaries and professionals) the opportunity to automatically receive competition results following the input of scorecard data. It is also intended that the system should calculate players new handicaps following their participation in a competition. [10] provides information on how to calculate a handicap in golf.

A database will be used to store all of the details involving both members and competitions. The desire is to have the back end database, which interacts with the presentation layer of this golf scoring information system. The application layer will provide this connection between the database and the graphical user interface acting as a communication pathway for querying the back end of the system. This will provide a means of access for the user, as they will be able to query the database and get any results they require sent to the presentation layer.

The system should also provide a means of secure access in the form of a user login so that only authorized personnel are eligible to access it. There will then be a main menu interface where all the necessary options are available to the user. Therefore whether they wish to add a member to the club or to search for a competition for example it can be easily done from this location. As far as the members are concerned this information system provides a means to search for members, edit their details, add members to the club and delete them if necessary. All of these different functions allow the user to have complete control of all of their clubs members at the presentation layer of this object-oriented approach to architecture. The user also has control over club competitions, as they are able to search, edit, add or delete competitions from the annual schedule.

There is also a scope to produce an annual calendar for the club professional so they can record any lessons booked with members. This will provide them with a weekly schedule so that they are able to

check when they're busy with lessons, they can also allocate time slots and remove any cancelled appointments. Furthermore, there is also the need for a membership fees mechanism, which alerts the user when members haven't paid their annual fees. Such a function would require this information to be stored in the database and a counter would need to be added in order to determine how many days over paying certain members are in order to trigger the alert to the user. However, this is an advanced requirement of the system so although it would provide additional functionality it is not an essential objective.

## 1.5 Project Requirements

### 1.5.1 Minimum Requirements:

- A database containing golf members information
- The system must allow data entry from scorecards through a user interface
- Output must include a list of competition results after data processing
- Must have the ability to deal with changes to the database through a user interface

### 1.5.2 Advanced Requirements:

- To incorporate a club professional schedule into the system so that members are able to book lessons on this calendar.
- To incorporate member's fees into the system so that the secretary has a record of who has paid their annual fees. Thus, providing a mechanism, which automatically generates emails to members whose fees are overdue.
- Allow for a points system where members get points for where they position in each competition they play. This could then be put to use where annual awards are given to decide who has been the most consistent player.
- An alternative approach to the point above is to maintain which players handicap changes by the largest margin over the year. This provides another helpful mechanism when determining who is most improved player for a whole season at a club.

- Introduce an eclectic scoring system so that each member knows their best round using scores from all of the competitions they have played.

### 1.5.3 Extensions On Advanced Requirements:

- Allow members to access the system remotely so that they are able to view weekly scores of competitions and edit their own personal details for their golf club.
- To incorporate an annual calendar detailing any social event that occurs at the club.
- To provide a desktop panel which acts as a platform for the graphical user interface.

Visit Appendix J for more detail on all the requirements stated above.

## 1.6 Relevance to Degree

In order to produce the desired solution I will be utilizing both the skills and knowledge that I have acquired during my time at Leeds University. In order to fulfill this there are certain modules that I have studied which are of great importance to this task, and it is from these that i need to build upon the principles taught to me. Such modules include database principles and practice (DB21), which helps with building the back end database and object oriented software engineering (SE20) which taught me the relevant skills to create an object oriented approach to architecture.

## 1.7 Project Structure

All aspects of this project are to be undertaken in parallel with the writing of the report. An initial phase is to outline project goals and apply research into competitive products, relevant methodologies and then analysis can be carried out on the problem domain. Once this has been completed the requirements will be identified which will be followed by the design phase. It is the intention that all of these sections described so far will be documented on completion. The implementation phase will follow and after this will come thorough testing and evaluation so that conclusions of the system can be made in the final section of this report. The methodology reflections and product assessments will then be used to document any possible improvements and enhancements.

# Chapter 2 – Background Research

## 2.1 Introduction

The following chapter discusses the background research carried out by the developer, which entails thorough investigation of all essential parts of this golf scoring information system. This includes an investigation of usability fundamentals which will be carried out so that the interface mediates between the user and the proposed computer system effectively. A decision will also be made involving which DBMS is most suitable for this application in terms of which will best fit this object oriented approach to architecture. A final discussion involving current technologies will take place, and within this, justifications of the selected programming language and software tool will be documented. This will be used to discover the most appropriate development tools to implement the proposed solution.

## 2.2 Usability

The level of communication between users and the system is of paramount importance; therefore the following investigates usability principles with the intention that the knowledge gathered from these can be used to establish criteria for the development of interfaces. In doing this the chosen usability features should be implemented so that they reflect the vision of this project. The following describes the relevant usability fundamentals to achieve this state.

### 2.2.1 Nielsen's Usability Principles

If a system is too complicated and complex in its general outlook and main functionalities users will not persevere with it. This relationship between the user and the system is built on the state of the usability fundamentals of the user interface. In order to enhance this relationship, Nielsen’s usability principles are an important guideline in achieving fundamentals like consistency and flexibility so that the user achieves an optimum level of performance from the system. Therefore these heuristics are deemed an essential part of interface design. Nielsen's usability principles can be found at Appendix J and they will all be taken into account when designing the interface for this golf scoring application.

### 2.2.2 The Principles Of Interface Design

As far as design principles are concerned there are five essential topics, which must be reflected in the proposed interface, so that the task of creating highly usable system is achieved. In addition to this they will also ensure that the users perceptual system receives information from the application itself as they will both operate inside an environment, which can ultimately affect system efficiency.

The first principle is naturalness, because a good interface appears natural. “It should adopt an instructional tone since it has been shown that users become discontent with systems that try to be personal” [19]. Furthermore, the system should adapt to the needs of the user and not expect the user to adapt to its needs.

Secondly, consistency is of extreme importance as the interface should provide this, “in its requirements for input and have consistent mechanisms for the user to make any demands on the system” [4]. In achieving this, the proposed interface will not have the user guessing why something has happened or what to do next which is a vital part of establishing an effective relationship between the user and the system.

Another key principle of interface design is relevance. This is because the proposed system should not ask for redundant material as minimum user input and minimum system output is necessary for any task the user wishes to address. This will maximize user performance and minimize potential errors. “It is easy to develop a system that frightens users; the hard part is building a system that makes them comfortable and content” [4]. This must be taken into account and put in to practice when designing the interface for this golf scoring information system.

A penultimate principle is supportiveness. This is because the interface should provide adequate information so that the user can operate and perform a task. The proposed system should therefore overcome this problem by providing sufficient and specific help in order to help users formulate input

requests. When addressing system support, it must be taken in to account that, “naive users need much more help than experts” [4].

The ability of the interface to be flexible is the final design principle to be discussed. This is essential to the proposed interface for this golf scoring information system, as it should, “accommodate differences in user requirements and also in user preferences and level of performance” [4]. An example of this could be that when the club professional accesses the system they are able to view their personal booking calendar for member lessons. This however will not be accessible to the club secretary who doesn’t require this information for their everyday use of the system. In achieving this personal state, it will provide a variety of support levels as the system, “gears itself towards the range of needs required by the different user types and their tasks” [4].

In addition to these design principles an essential part of creating a good user interface is getting the users attention. Therefore techniques involving intensity, marking and size issues will play a vital part in creating this relationship between the user and the system. However, “there is also a danger in creating cluttered displays by overusing these techniques” [19] and this will also be accounted for in the design phase so that this and all of the sections previously discussed are in harmony with the goal of simplifying the users task.

## 2.3 System Methodologies

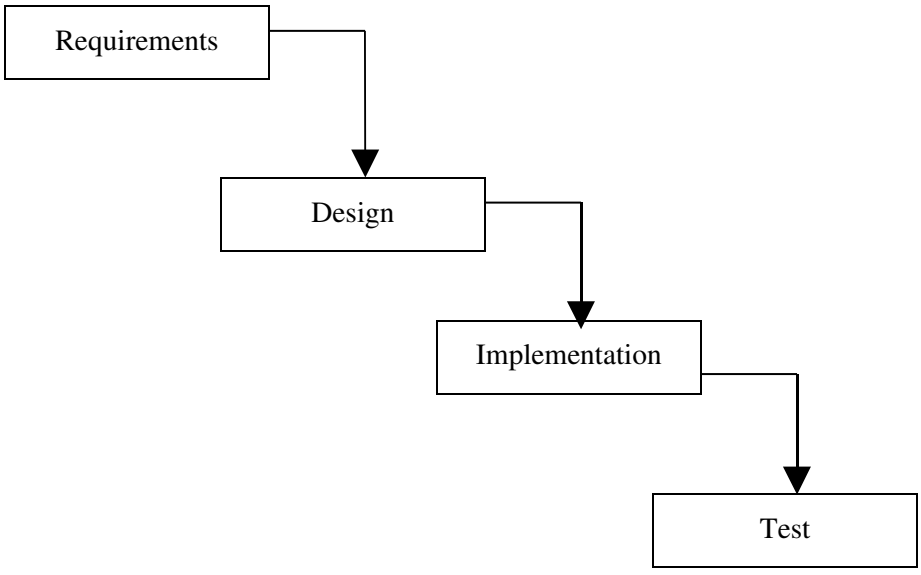
As far as choosing the correct methodology is concerned it is vitally important that this matches the systems criteria. In software engineering there is an implicit assumption that projects move from task to task with a purpose in mind and with someone in control. In order to provide an optimum solution for the problem situation it is necessary to select a methodology, which is appropriate to the organizational context because this selection is based on specific intellectual traditions and will therefore seriously reduce the risk of project failure.

The Contingency Framework from [13] notes will also be investigated in order to aid this decision as this model can be used to help select an appropriate methodology for a given organizational context. This involves searching for the appropriate requirements and processes in order to determine which methodology has the best fit for this system development. As far as the methodologies themselves are concerned, there exist three general paradigms, which are the Waterfall Approach, Iterative Development and Component Based. The following discusses each of these along with various methods inter-connected with them.

### 2.3.1 The Waterfall Approach

This incremental approach consists of four stages of development, which forces a precise definition of requirements and aids avoiding errors and reworking. The diagram below demonstrates how this method works.

Figure 1.0 The Waterfall Model




This approach bases itself upon the idea that once each stage in the life cycle is complete then it is signed off and development then goes onto the next stage. In forcing the developer to think about what they are

going to build and then planning for how this idea should be implemented this methodology has an extremely powerful concept. This disciplined approach includes a variety of useful analysis and design tools and is therefore used by many companies. For further information on this model see [1].

### 2.3.2 Iterative Development

All activities in this methodology are interleaved with each other, where an initial system becomes rapidly developed from high-level specifications and this is then refined over subsequent iterations. This method forces a team to focus on the most important issues, which are most critical to the project. In addition to this serious misunderstandings are made evident early in the projects life cycle and continuous iterative testing enables an objective assessment of the projects status. Furthermore, inconsistencies between requirements, designs and implementations are detected early. Additional information on iterative development can be found at [23].

### 2.3.3 Component Based Development

This methodology assumes the use of some pre-built components in order to aid further development. This focuses on architecture and the integration of components rather than building from scratch. An example of this is prototyping, which provides a base product to be developed further. There are three different approaches to this method, which are:

1. Throwaway prototyping: Designed with the intention that they will be disregarded after analysis.
2. Evolutionary prototyping: An initial prototype is constantly modified throughout the development.
3. Incremental prototyping: The process of delivering a new prototype each time the old one has been analyzed and any improvements have been identified.

When using this prototyping technique, risks are eliminated like users not being happy as well as the problem of the solution not working. In overcoming this problem with UI prototypes and proof of architecture, component based approaches aid a low risk solution for development. Although this

technique helps users gain familiarity with the system prior to its release, it can also lead the developer to miss deadlines. The developer could also place too much of an emphasis on each prototype which may lead to more important aspects of the project, like the completion of any documentation which could be overlooked. Component based development can be found in more detail in [1].

### 2.3.4 Methodology Conclusions

After assessing all of the different frameworks available it has been decided that the most effective methodology for this project is a mixture of these discussed. Although each method provides various advantages, there exist to many drawbacks to each approach and so this decision cannot be based solely on one of them. Therefore, from these investigations and studying the Contingency Framework it has been decided that a Unified Process, which repeats the Waterfall Model quickly over a number of iterations, is the most appropriate approach to aid development for this application. Evidence of these iterations exist during implementation in Chapter 4.

In using this method, there are four distinct phases for development, which are inter-connected in this spiral approach to development. [13] states these are inception, which addresses any architectural risks. The second of these is elaboration where requirements and design risks are addressed. Thirdly, the construction phase takes place in a less risky project environment. The final phase involves transition, which is reasonably risk free. All of these base themselves on the concept of using resources accordingly against time. “The Unified Process is a design framework which guides the tasks, people and products of the design process” [11]. As far as development is concerned, it provides answers to who is doing what, when they do it, how to reach a certain goal and both the inputs and outputs of each activity. This method is both iterative and incremental which as previously mentioned does not attempt to complete the whole design in one go. Also any prototypes that are developed will be used to explore some aspect of the design process and so this technique involving much iteration doesn’t base itself on rapid prototyping. A drawback to the Unified Process is that it requires additional planning to those approaches discussed. However, this has also been recognized as a potential advantage in the long run as the end result is that the

developer will incrementally produce the system being designed whilst identifying any risks to the process.

The Unified process was developed by Booch, Rumbaugh and Jacobson and offers a hybrid approach to object oriented analysis and design. Unified Process implies using tools like UML, as this is a use case driven object oriented approach to architecture. This use case driven approach ensures that any primary requirements of the system are identified and that the evolving design is always relevant to what is required. It was explicitly designed to work with UML and was therefore developed in tandem with it. An additional reason for choosing such a technique is that, "UML has become the de facto standard," [11] and has therefore been adopted by the Object Modeling Group and by all vendors of object modeling tools. "UML is made of a number of models that together describe the system being designed where each model comprises one or more diagrams with supporting documentations and descriptions" [11]. Using such a method means that the following diagrams will determine the outlook of the system.

1. Use case diagrams - present the interaction between users and the system.
2. Class diagrams - present static structure of the system.
3. Object diagrams - present objects and their relationships at a particular point in time.
4. Activity diagrams - describe the flow or typical activities of tasks.
5. Collaboration diagrams - show the sequence of messages between collaborating objects for a particular task.
6. Component diagrams – illustrate the physical structure of the code.

Therefore, "UML is a notation for visualizing, specifying, describing and documenting a software system" [11]. One of its key aspects is that each diagram should be consistent with any other diagrams, which represent the same information. More information about UML can be found at [20].

As previously mentioned, although the Waterfall Approach provides a disciplined approach for development, it doesn't overcome problems where reworking is necessary as each stage is signed off on

completion. Furthermore, a prediction that one phase can be virtually completed before the next has begun is valid in an environment where this is the fourth or fifth development project to be built in the same domain. However, it would be foolish to rely on such a prediction for this application especially because it is the first development project. The iterative approach is not appropriate for this development because it bases itself on recognizing risks, errors and the possibility of failure in addition to it being inappropriate for safety critical systems. This method is therefore far too honest for comfort as this development focuses on the Ostrich attitude to risk, where risks tend to be ignored.

## 2.4 Database Technologies

This section contains discussion not only of the relational database itself but also various database management systems. The latter will investigate many different technologies along with both advantages and limitations to each of these so that a conclusion can be made determining which DBMS is most suited to the requirements of this golf scoring application.

### 2.4.1 Fundamental Requirements of the Database:

- Allow add, remove, edit and update functionality.
- Provide both a fast and reliable service.
- Protect data from unauthorized users.
- Allow queries to extract or manipulate data.
- JDBC driver must be used for the connection.

### 2.4.2 Database Management Systems

There are many DBMS' available to choose from which can be utilized for this golf scoring system. This decision must reflect the nature of the application. The following discusses the main technologies available with the intention of choosing one for this object-oriented approach to architecture. These DBMS' include:

- **Postgresql** - This is a sophisticated open-source Object-Relational DBMS supporting almost all SQL constructs. Run under the cygwin emulation, this is making progress in performance and stability. Having foreign keys, views, sub selects, and transactions can all be very attractive in this. This also copes well with cross platform compatibility.
- **Oracle** - They supply a range of software to manage, share and protect data and information. This holds up very well in super critical environments as well as having the ability to manage effectively under a large number of database accesses.
- **Microsoft SQL server** - This is a DBMS produced by Microsoft. It supports a superset of Structured Query Language (SQL), the most common database language. It also provides nice development tools.
- **MySQL** - An open source RDBMS that uses SQL, the most popular language for adding, accessing, and processing data in a database. Because it is open source, anyone can download MySQL and tailor it to their needs in accordance with the general public license. It has a large user base, therefore the code is well tested and has wider use in production environments. It can therefore be used on various operating systems.

### 2.4.3 DBMS Conclusions

It has been decided that Postgresql is the best-fit DBMS for this type of system because it has many advantages over its competitors, which are related to the system environment. The following demonstrates this decision more clearly by comparing Postgresql with each other DBMS individually.

Postgresql and MySQL

- Postgres does a very good job supporting referential integrity and MySQL has some basic provisions for this.
- As previously stated, having foreign keys, views, sub selects, and transactions can all be very attractive in Postgresql and this is a desirable to have in the system.
- Where security is concerned, Postgres can limit logins based on different criteria.

- Postgres is more standard compliant.

Postgresql and Oracle

- Oracle is expensive to purchase whereas Postgres is free of charge.
- The database isn't going to be constantly fetching data, which is a strength of Oracle that isn't really relevant to this system.
- Postgres works much faster than Oracle.
- Postgres is more standard compliant.

Postgresql and Microsoft SQL server

- The Microsoft product is expensive to purchase whereas Postgres is free of charge.
- For many uses, Postgresql is just as suitable as Microsoft SQL Server, but with a big cost advantage.
- The advantages of Postgresql are cost and the ability to look at the source code to understand what's going on.

Further information on Database Management Systems can be found at [15].

## 2.5 Programming Languages

Many programming languages exist which could be chosen to produce this golf scoring system. The following discusses the Java and C++ programming languages and a decision will be based on which is more relevant for this task.

### 2.5.1 JAVA

Java is an object-oriented programming language developed at Sun Microsystems who currently maintain and update Java regularly. [25] Explains that Java generally refers to a combination of three things:

- The Java programming language - A high-level, object-oriented programming language.

- The Java Virtual Machine - A high-performance virtual machine that executes byte codes on a specific computing platform.
- The Java platform - A JVM running compiled Java byte codes, usually calling on a set of standard libraries such as those provided by Java Standard or Enterprise Edition.

### 2.5.2 C++

C++ is another object-oriented programming language that is viewed by many as the best language for creating large-scale applications. Java is based on C++ but optimized for the distribution of program objects in a network such as the Internet. C++ maintains aspects of the C programming language, yet has features, which simplify memory management. It is also one of the most popular languages for graphical applications. Additionally, " some of the features of C++ allow low-level access to memory but also contain high-level features" [18].

### 2.5.3 Programming Language Conclusions

As previously mentioned, the decision of choosing one of the two languages described above is based on the purpose of this system. Although C++ is often a popular choice for graphical applications it has been decided that Java is the more appropriate choice for this development. Not only does Java provide better portability, as it is platform independent, it is also more secure to use than C++. "Java is supported by thousands of classes that simplify remote networking, GUI/Windows, and client side interfacing with enterprise servers" [25].

Another feature which proves Java is the more appropriate to use, is that it is less complicated than C++. It also supports built in features like memory management and concurrent processes, which are major obstacles in C++. This effectively means that you make fewer errors, which increases productivity.

Despite all of these advantages of Java, it has a major speed disadvantage compared to C++. This could be vital with large-scale development projects but this development is far smaller than that involved with

some commercial products. Therefore, this is not deemed to be to much of an obstacle because of the intended project size.

## 2.6 Development Tools

Although many tools exist for coding, it has been decided that this task requires an editor, which provides some additional functionality to that provided, by simple applications like Textpad and Gvim. Following the decision of which programming language shall be used, it is necessary to use an appropriate text editor, which can best aid this development. The following discusses two of these, so that a decision on the intended software tool can be made.

### 2.6.1 Eclipse

The first of these tools, which will be discussed is Eclipse. This is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the life cycle. [22] explains that, "Eclipse provides consistent formatting, intuitive debugging and JUnit support, class and package summaries and hierarchies." A hallmark of the Eclipse user interface is that it makes extensive use of wizards, such as creating new projects or classes and connecting to a CVS repository. It is the intention that the development will make use of CVS in order to code from various locations. It also eliminates a lot of repetitive and trivial work, leaving more time to think about the code itself and algorithms involved with this.

### 2.6.2 IntelliJ IDEA

A second tool which could be used for development is IntelliJ IDEA. This is an intelligent Java IDE, " which is intensely focused on developer productivity that provides a robust combination of enhanced development tools" [22]. The users and third parties continuously extend its functionality. IDEA uses swing/AWT which gives Eclipse the edge in performance as this tool uses SWT. However, on a reasonably powerful machine, performance issues between the two are negligible.

### 2.6.3 Development Tool Conclusions

From examining both of these tools, it has been decided that Eclipse will be used for the development phase of this project. A reason for this is that the user interface provided with Eclipse makes tasks slightly easier because the appropriate tools are close at hand. “It is designed using the concept of a perspective, a collection of views, editors and other UI elements that are best suited for performing a specific task” [22]. Another important difference between the two development tools is that Eclipse is an open source project therefore it is free to use whereas IDEA requires a license for use. In conjunction with this, Eclipse’ open source nature has attracted a large community of developers producing plug-ins which can be integrated into this project.

Although Eclipse is the correct choice for this development it has been recognised that there are some disadvantages of this tool compared with IDEA. The Java language level support is far better with IDEA and the IDE based code assistance works much faster. In addition to this, the whole configuration process with IDEA is much nicer designed and a lot easier to accomplish. Finally, although IDEA is not a platform based on plug-ins, its much easier to find and install plug-ins using this tool.

# Chapter 3 – System Design

## 3.1 Introduction

The aim of this chapter is to produce an effective design to overcome this golf scoring systems problem. It is imperative that the application design uses all the knowledge gathered from the analysis and background research so that all of the proposed functionality directly maps on to each requirement of this system. In doing this all problem areas will be covered, and therefore if the implementation phase is written in conjunction with the system design, a working solution should be the result. The design process will be split into sub sections of linking the main functionality, the database design and finally the layout of the graphical user interface.

## 3.2 Systems Architecture

Figure 3.1 below demonstrates the fundamental organization of the system and the relations between these parts.

Figure 4.1 System architecture

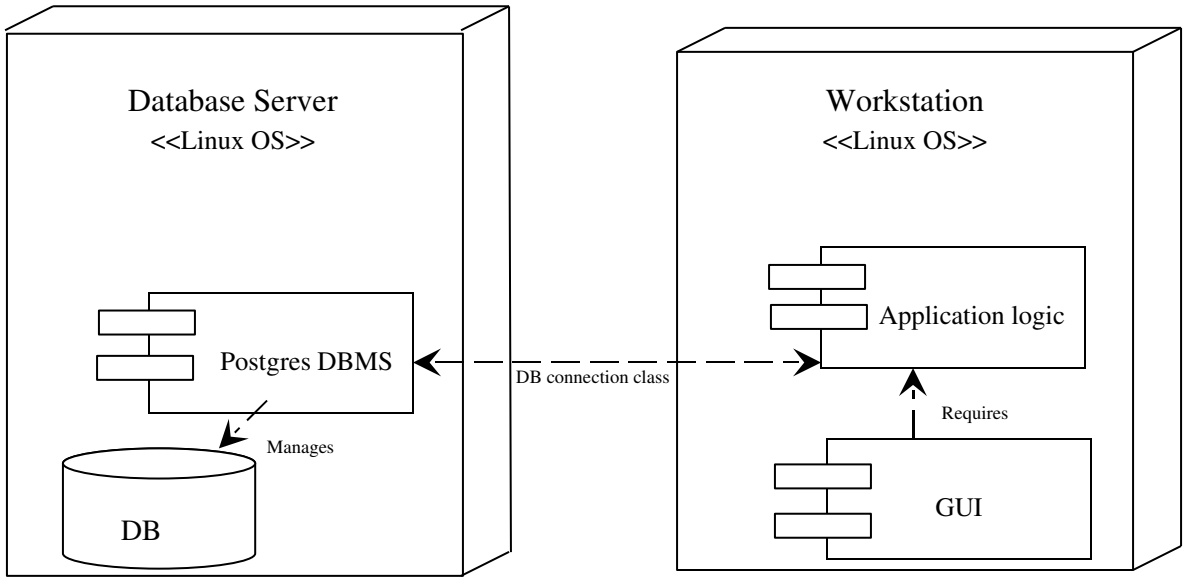



Figure 3.1 shows how each part of the system is intended to fit together. The database will be managed by the Postgres database management system. Therefore any changes to the database will be executed by this. All connections to the database are made through a DB Connection class which will set things like the driver name and database location. The application logic is used to manage the user input through graphical user interfaces. This logic performs many operations like calculating new handicaps or working out competition results. It therefore covers all of the necessary functionality stated in the objectives in 1.2.

## 3.3 Incorporating system functionality into design

The hardest part of the design process is deciding on the different functionality and how all of this fits together. Therefore, UML is used to help overcome this task as producing a use case diagram not only

describes the main functionality but is also a method of discovering how parts of the proposed system will fit together. Appendix H demonstrates the registration process for this system.

## 3.4 Competition Process

Now that the functionality of registering users, members and competitions has been outlined it is necessary to focus on the competition processes as far as entering scores and calculating results is concerned, after all the proposed system will be built around this mechanism. Appendix I demonstrates a work flow of events using swim lanes in order to discover the entire process, which a user of the system will go through when dealing with the scorecard data.

## 3.5 Database Design

In order to aid the database design, a concept level class diagram will be implemented to discover all of the classes and associations involved in producing this solution. These can then be used to discover tables and their fields required in the database. [20] agree with this technique because, “the way in which an association is implemented may be guided by the information represented on a class diagram for an association, but the implementation does not necessarily need to be ruled by that information.” It is for this reason, that the concept level class diagram will only be used as a guide with the database design.

Figure 3.2 Concept level class diagram

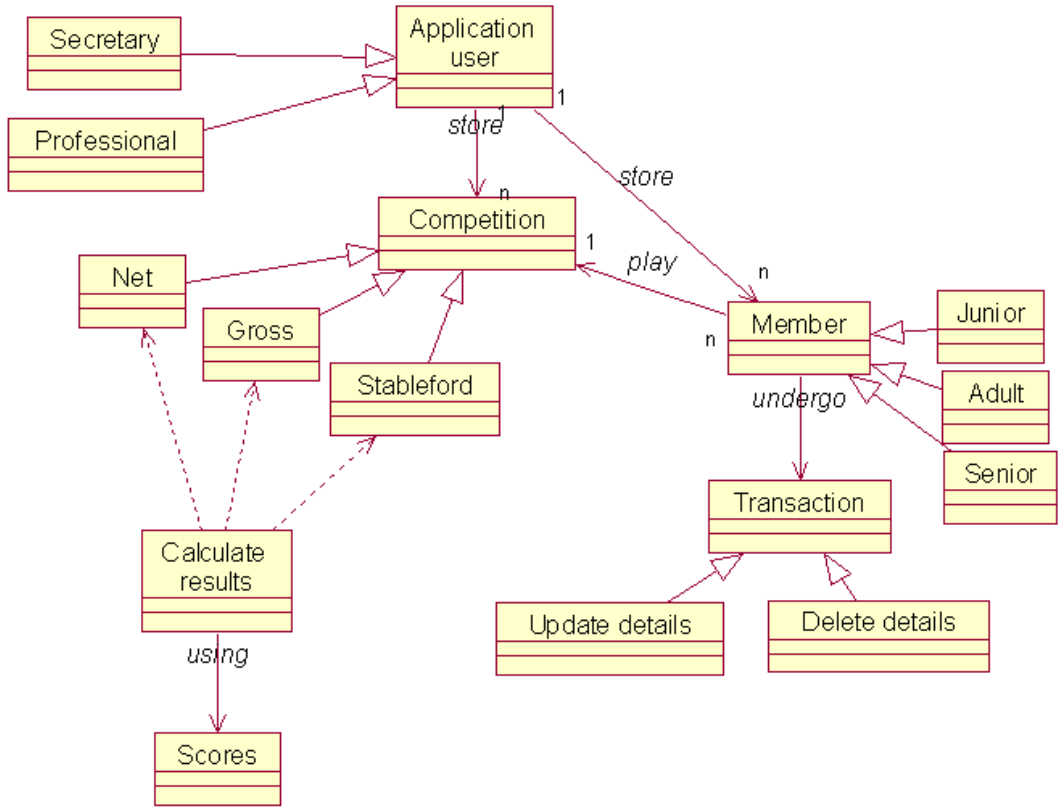



Figure 3.2 demonstrates logic covering all requirements of this solution. It is from this diagram that the following tables can be proposed, which are designed to hold data so that all the main functionality can be later produced through the application logic.

- Member details with primary key member ID.
- Competition details with primary key competition ID.
- Course Information.
- Hole Information with primary key stroke index.
- Scores contains two foreign keys (member ID and competition ID).
- Users details with primary key username.

To see the complete contents of each table visit Appendix E.

In order to fulfill all of the requirements, there must also be constraints to limit certain capabilities. Each of the following issues therefore requires a constraint so that the complete solution works effectively and is realistic enough to be used in a golfing environment.

1. A member must not be able to play a competition twice.
2. The first time the system is used both the login details and password need to be accepted.
3. Scores must be calculated based on the competition type.

### 3.5.1 Search Queries

It is intended that there will be many different types of queries used for this application, because the application logic requires the use of all data in the database, when addressing numerous tasks to fulfill the applications requirements. The following outlines each of these queries, which need to be implemented, as well as the purpose they are being used for.

- Insert query – This will be used to add data into different tables in the database. It will be called upon where a new members, competitions or users details are input into the application. Furthermore, it will also be required where scorecard data is input so that each member's results can be added to the scores table in the database.
- Update query – This type of query will be called upon when data needs to be modified in the database. As far as this information system is concerned this will be required on occasions where either members or competitions details have changed and therefore this change needs to be replicated in the database.
- Delete query – If a member fails to renew their membership or it is decided that a competition is to be removed from the database then the delete query is required. If a competition is removed then all the scorecard data associated with that competition will also need to be deleted as well.
- Select query – This will be the most commonly used query by the application logic for this solution. As discussed later in the development of the application, the calculation of results requires many select queries in order to obtain the appropriate information for this task. Additionally, these queries will further be used where member and competition details are

displayed by their user interfaces. They will also be required during a user search for either a competition or a member where they specify criteria and some keywords, which in turn produces a list of matches from this query.

- Count query – This count function will be used in conjunction with select. It will be required on occasions like the user login where it will check whether the number of instances of the provided input is equal to 1 or not. The application logic can then use this result in order to decide on whether the user can access the system or not. However, using this approach alone is not a very secure method for a login procedure. Therefore, in order to seriously reduce the success of an SQL injection it is necessary to use regular expressions with this procedure. This will be used such that only certain characters are eligible for input on the login interface producing a more secure system.

## 3.6 User Interface

The design of the user interface must abide by the usability principles outlined in the background research chapter when considering features like colouring, navigation and graphics. [16] states, “The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible.” If the user interface is to achieve this, it needs to provide a high level of novice learning which according to [16] “will also be good for the experts.”

### 3.6.1 Colouring

The choice of the colour scheme must reflect what this project aims to accomplish. The task of initially choosing primary colours for this solution, must bare in mind that users will be looking at these interfaces for long periods of time and therefore choices shouldn’t be made which could put strain on a users eyes. The primary colours defined will then be used to discover complimentary secondary colours, which are simply shades of these initial choices. These will be used to accent the primary colours, in cases like highlighting words or titles on an interface.

Using these in conjunction with primary colours aims to draw the users attention to pieces of important information. A final note on colouring is that they are, “deeply associated with feelings and interpretations on just about everything we see” [14]. Taking this into consideration, “ the colour choice mustn’t say the wrong thing” [14]. To overcome this problem choices must therefore be fresh and unique providing a combination which users find effective.

### 3.6.2 Navigation

The navigation of an application is extremely important if the system is to have a high level of usability. [16] comments on the breadcrumb trail where, “users should be able to use a navigation system to make their way up the systems hierarchical structure to the menu of the current section, and to any section menu above that, and ultimately to the main menu.” This will be strictly followed using the quick links panel, which contains all of the main processes, which the user will carry out. Although this isn’t the traditional way of using the breadcrumb trail which is normally comprised of, home > main section menu > subsection menu > this page, however using this panel will allow the user to navigate anywhere at anytime. Furthermore, the main menu will exist until the user decides to close down the system so this can also always be used for navigation.

Another navigation tool, which will be used, are station signs. They are closely associated with the breadcrumb trail and their purpose is to alert the user wherever they are throughout the application. “Ever woken up when your train has pulled into a station and wondered where you are? You expect the station signs to tell you. That's why they're there” [21]. Effective navigation will use the breadcrumb trail and changes the colour of a current section in any navigation bar covering the main sections.

### 3.6.3 Graphics

Although there will be few graphics evident throughout the application, this is also an important principle in achieving a usable state. “Interfaces represent the interactions, intentions, goals and tasks of users,” [6] and so greater success is gained from putting the usability before the graphics in this design process. The

system will be limited in this area to a logo, because it is intended that it shall be both intuitive and sensible as this field of software development is dictated by user actions rather than visual appearance. However, the inclusion of this tool aims to balance the application being both visual and functional. Appendix K contains a proposed GUI layout.

# Chapter 4 - Development of the Solution

## 4.1 Introduction

Using information gathered from the design of the system, this chapter discusses the different phases, which make up the implementation of this golf scoring solution. In order to best replicate the UML from design, it was decided that the proposed system should be broken down into smaller components with the intention that they would later be pieced together producing a complete solution. It was therefore essential that each of these components represented a different aspect of the minimum requirements so that the main functionality of the application could be completed first.

The components were prioritized such that phases could be pieced together on completion, which allowed for the testing of the solution in small proportions, rather than doing this at the end and discovering lots of errors. Once the minimum requirements were completed, with time abiding some further coding was done in order to further enhance the system thus providing a niche with this software in comparison with existing solutions, this is mentioned later in the chapter.

## 4.2 Component: Database

The initial design of the database was made up of 6 tables, which were intended to store data covering every aspect of the intended solution. These included:

1. A member's details table containing personal details so that they can easily be contacted by the golf club. It also stores each person's member type, allocates an ID, and records his or her current handicap.
2. A competitions table containing all information related to a specific competition. This includes data such as competition type, which is later used when calculating the results of a competition. This table also stores the date of the competition, allocates an ID which is later used as a reference and records the name of the competition.
3. A hole information table, which stores data on each hole including the name, par. This data is also used to when calculating competition results.
4. A course information table that stores the name of the golf club and records the par from those 18 par values in stored 3.
5. A users details table containing the type of the user as well as their specified username and password. These values are used when deciding whether a member is eligible to login to the system.
6. A scores table, which holds input values from scorecards. Therefore it contains every member's score in every competition. Each set of 18 results also contains a competition and member ID so that they can later be extracted to calculate results.

### 4.2.1 First Iteration

After coding all the above using Postgresql each of these tables were then added to the database sitting on the database server. Following the completion of this, the application needed a way to communicate with the database, in order to later process transactions, which add, modify, update or delete data. Therefore a connection class was introduced to provide this gateway. This required information including the driver name and the location of the database in order to produce the connection. It was intended that every time a call was made to the database, it must create an instance of this connection class to do it.

### 4.2.2 Second Iteration

As mentioned in 2.3.4, a task can be iterated a number of times until completion, therefore a second iteration is required to complete this task. Following the implementation of the database it became apparent that there was a field missing within the hole information table. In order to successfully implement the calculation of competition results, it was vital to not only provide a solution which used the clubs hole information with regards to pars, but it was realized that the Stableford competitions rely on the stroke indexes of holes with their calculations. Therefore as part of this second iteration this feature was included in the hole information table in conjunction with the par of that hole so that this feature of the system could later be implemented. In doing this, it was also necessary to include a primary key using this feature because each hole on a course has a unique stroke index.

## 4.3 Component: User Interface

Following the database setup was the creation of the user interfaces. As previously discussed in 2.2.1, it was decided that each user interface must strictly follow the guidelines provided by Nielsen to produce a solution with maximum usability. Furthermore, it became evident which interfaces were necessary to fulfill the minimum requirements using the UML activity and use case diagrams obtained from the design phase. Therefore the following principles required interfaces providing interaction with the user:

1. A main menu with all of the vital functions including adding members, competitions and users. In addition to this there must also be a gateway to search members and competitions from this menu.
2. An interface to add course information must be provided with appropriate validation so that all hole data and club information can be entered into the system. Providing such a feature means that the application can then be personalized for a specified club. An example of this is with the scorecard provided which is implemented so that the names of each hole appear on this interface. Additionally, it is also vital that this data is entered because it is these values (hole pars and stroke indexes) that are used to calculate how well players have done in competitions.

3. An interface to add member's details must be provided with appropriate validation so that a member can be added to the golf club. This must also be implemented for users so that a new user to the system is able to gain access with their details.
4. An interface to add competition details must be provided with appropriate validation so that a competition can be added to the golf club for which members can participate.
5. In order to be able to edit or delete a member's details there must be an interface, which takes input and provides a search following some criteria selected by the user. There must also be a user interface searching competitions for the same reasons.
6. Following on from the previous point, once a user selects a member or competition there is a need for an interface, which displays the information based on this search. This information can then be modified using this interface.
7. It is intended that from the competition details interface described in 6, that there is navigation provided which allows the user to enter member's competition results through a scorecard interface. On the completion of each set of results the user must then have the option to either clear or add this data to the player scores table in the database.
8. Once all of the competition data has been added to the scores table the user must then be able to view the results of the competition again from the interface described in 6. These results will be calculated depending on the type of the specified competition because they're calculated differently for each. From this list of results the user must then have an option to update the handicaps of members based on how well they performed in the competition. It is also intended that the user also has the option to print this list which can then be pinned up in the club house for the members, however this option wasn't stated as part of the minimum requirements.
9. A final interface, which must be included to successfully fulfill the minimum requirements, is the introduction of a user login. These details will be compared with those entered for a new user so that only eligible users can gain access to the system. It is the intention that for the first time the system is used, the initial username and password details entered will provide system access and be stored in the database because there will be no other records to compare these values against.

As previously mentioned in the design chapter, in order to further the usability of the application it was also decided to include a quick links panel on many of the interfaces achieving consistency throughout the solution. This feature provides a more efficient way of navigating through the application because the user won't be required to constantly return to the main menu in order to carry out another task, as all of the main functions will be provided for them in this panel.

### 4.3.1 First Iteration

Development of the Main Menu Interface

As far as the main menu is concerned it only required one iteration for completion. Using the design it was already known what needed to be incorporated within this. As this is the first interface a user will see when gaining access to the system, the layout and colour schemes were deemed to be very important. As previously mentioned, "It is easy to develop a system that frightens users; the hard part is building a system that makes them comfortable and content" [4]. Therefore it is vital that the features used achieve consistency throughout the application so that users are comfortable with the system.

In developing this user interface, an external package, Jgoodies was used to aid the alignment of the various buttons and overall layout of the form. It was also deemed necessary to personalize this interface with the intention that this feature would make users more comfortable with the application. This personalization included the chosen user name by the logged on individual being displayed on this form beneath the date. This also overcomes any problems of identifying which user is logged on at any time.

Development of the Add Member Interface

In order to develop a consistent solution throughout it was essential to implement all of the interfaces, which required some data input (like when adding a competition or a user) in much the same way that this first interface for adding a member was produced. As previously mentioned a quick links panel was to be included in many of these interfaces to improve the system navigation therefore this became integrated

into this interface for that very reason. The title of the main menu interface discussed above was added to a box using a white border and this was again used when developing this interaction tool.

As far as the layout of each field is concerned, this was again made easier using Jgoodies to align fields in such a way that input could be made easier. The buttons were aligned horizontally towards the bottom of this form to keep this consistent with many existing applications. Many of the interfaces receiving input were developed in this way using four distinct panels where the title panel lies north, the buttons panel lies south, the quick links panels lies west and the main data panel lies east on each form. See Appendix F for the user interface work flow.

## 4.4 Component: Back End Classes

Following the creation of the user interfaces, the main functionality of the system had to be coded in back end classes. It is in each of these that the process of each button throughout the forms is calculated. Each back end class carried out all the functions involved with a specific part of the system. For example, the back end class for competitions involved updating, deleting, and adding records as well as calculating results for a specified competition. This was used by all the user interfaces, which were involved with competitions, and it is this application logic, which creates a connection between the user interfaces and the database.

### 4.4.1 First Iteration

The main aspect of this first iteration at this stage in development was the process of adding, updating and deleting a member, competition and user from the database. Additionally this phase also included the coding of correctly adding score data inputted by the user into the database. The application logic behind each of these was simply gathering the data inputted, converting it into an appropriate form (everything was converted to lowercase as this later helped with searching) and executing an SQL query using the manipulated input values. Furthermore, this iteration stage was also used to develop the search process of finding a member or competition, given some input and criteria, and viewing their record. This logic

however was slightly different because rather than simply modifying data in the database, this concentrated on returning some results which match the SQL query manipulated by the users input.

### 4.4.2 Second Iteration

Now that all of the basic functions had been completed it was time for the main functionality of the system to be developed. As previously mentioned, it was always the intention that once the user has inputted some scores into the scorecard interface, the application logic should then be able to work out values from these and produce a list of results depending on the type of competition the members have played in (whether this be net, gross or stableford). In order to carry out this process, the calculations needed to be completed on every set of results from a specified competition. Therefore in order to successfully work out each value, both the course par and the par of each hole would need to be used to compare each result set with these values.

In addressing the task of working out results from a gross competition it is necessary to simply find the total score hit by a member by adding all of their scores on each hole together. The only difference is with this and calculating results for a net competition is that the players rounded handicap is taken away from this total score to give a net score. However, calculating a handicap change is far more difficult than this. See [8] for details on how to work out this value given a players competition results.

### 4.4.3 Third Iteration

As a result of the second iteration it was soon realized that in order to produce a list of results for the stableford competitions, it was necessary to make comparisons of each net score with every stroke index. Therefore, as previously mentioned, the stroke index' were only stored in the database on the second iteration and so this is why these results couldn't be calculated in the same iteration as the gross and net scores. This calculation is based on a points system where players gain points for how they score on each hole in comparison with the par on each hole. The algorithm which calculates the stableford results can be found in Appendix G.

## 4.5 Component: Star Members Advanced Requirement

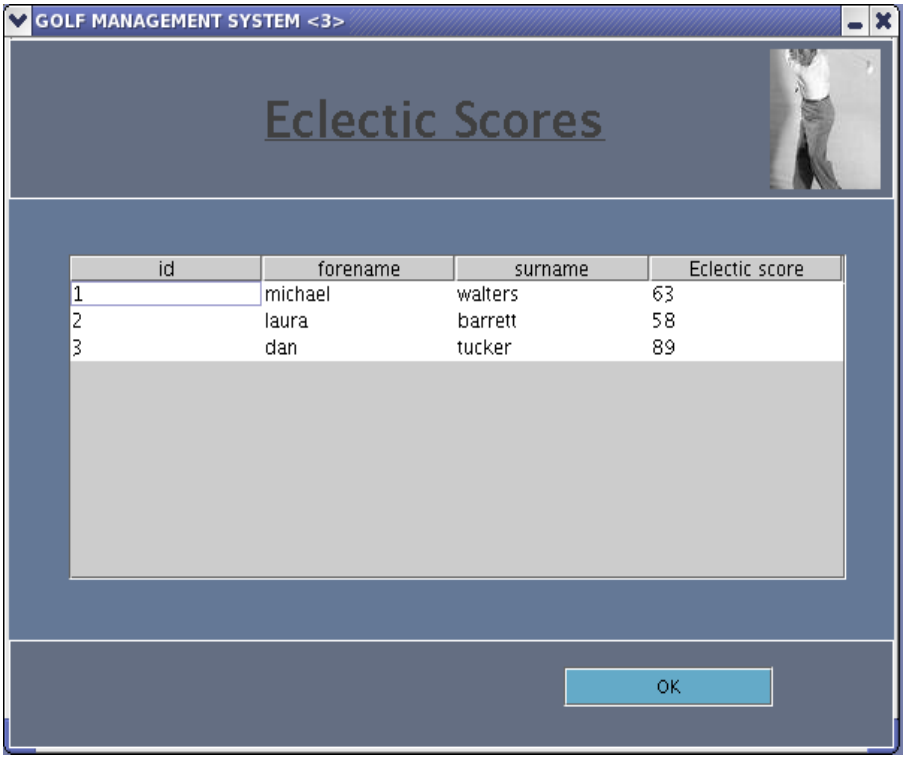
Following the completion of the minimum requirements as described above there was time to further develop the system as development was slightly ahead of schedule. Taking background research into account, it was decided that the best use of this time would be to create a star members feature because this has not been seen in any existing solutions. Not only does this provide a cutting edge to the main functionality but also as a member of a club myself it would be of interest to see which members are improving the most and shooting the best scores. All of the data used for this already lies in tables in the database it was therefore just a matter of extracting the correct data for this star members purpose. It was deemed important users could extract information like knowing the top scores and the players that have posted these, as well as finding the most improved junior member over a period of time. This could then be very useful when deciding on prizewinners at annual awards for example.

Figure 4.1 Star members interface

Another feature within this is the calculation of each member's eclectic score. The logic behind this calculation is that it takes each player's best score on every hole in all competitions and finds their best

gross score. This is another interesting feature, which could be used when giving an award. This not only encourages members to play the game more, thus further enhancing their skills as golfers but also provides a more competitive edge to competitions, as I'm sure all members would wish to hold the lowest eclectic score at their golf club. As far as this is concerned the user interface has been set up like many of the others where the panels are placed such that the main information lies in the center. This is shown in figure 4.2 below.

Figure 4.2 Eclectic score interface



# Chapter 5: Testing

## 5.1 Introduction

This chapter focuses on testing the system in a variety of different ways, in order to discover whether a successful working solution was produced from development. Testing involves evaluating the systems compliance with its specified requirements. Therefore in achieving this, the testing process will be conducted in two sections. The first of which will be compatibility testing where the software will be run on two different operating systems. Basic system tests will follow to ensure that system functions are working accordingly for the purpose they are intended to serve. An example of a system test is the result calculations from competitions, which will be later discussed in this chapter.

## 5.2 Compatibility Testing

The application will be tested on the following two operating systems with the intention that it is compatible with both. If this isn't the case then the systems build file must be adjusted in order to gain the desired level of compatibility.

### 5.2.1 Windows XP Home 5.1.2600

In order to carry out this test, the system files were put onto a CD and from here the software could be installed onto an SOC windows workstation. Following installation the initial interface was the login screen and from here access could be easily gained into the system once the correct user name and password were entered.

### 5.2.2 Fedora core 6

The development process was implemented using eclipse on a school of computing Unix workstation. Therefore, instead of running the system with this software, it was decided that the program should be able to execute from the command line in a terminal. In order to carry out this process, I moved into the

workspace directory where eclipse stores these files for the application and executed the login file from here, where the system had no problems running.

## 5.3 Basic System Testing

As previously mentioned during the introduction of this chapter, the purpose of this type of testing is to ensure that system functions are working accordingly. In order to achieve this, the basic testing will be comprised of result calculation, login, competition constraint and validation testing. Covering all four of these areas provides tests of all the main areas of the system, where it is intended that any faults discovered will be fixed so that the result of this investigation is a complete working system.

### 5.3.1 Result calculation testing

In order to test the methods implemented which calculate the results of various competitions, it is perhaps necessary to provide some test data, which will be input into the application to judge the reliability of the program. The following data table has been comprised with expected results so that the test values produced from the application can be compared against these.

Hole data

The data used in this table is intended for use with all of the following tests. The details of each hole have been recorded for use when calculating adjusted the scores in cases where a member has scored more than two shots over par on a hole.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Par | 4 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 5 | 4 |
| S.I | 1 | 15 | 5 | 6 | 17 | 13 | 8 | 12 | 7 | 16 | 2 | 18 | 14 | 3 | 7 | 11 | 10 | 4 |
| | | | | | | | | | | | | | | | | | | |
| 1 | 5 | 4 | 5 | 5 | 5 | 4 | 4 | 4 | 5 | 3 | 3 | 2 | 3 | 4 | 5 | 5 | 6 | 4 |
| 2 | 8 | 6 | 6 | 6 | 5 | 4 | 5 | 5 | 4 | 7 | 7 | 5 | 4 | 4 | 5 | 8 | 7 | 3 |
| 3 | 8 | 10 | 7 | 5 | 4 | 5 | 6 | 7 | 9 | 8 | 7 | 10 | 8 | 8 | 6 | 8 | 6 | 6 |

Gross score calculation test

The logic behind conducting this test is to compare the working of calculating the gross score by the system with some expected results in order to test how accurate the coded methods actually are.

| Test | Handicap | Expected gross score | System gross score | Expected handicap | System handicap |
|---|---|---|---|---|---|
| 1 | 12.9 | 76 | 76 | 10.2 | 10.2 |
| 2 | 26.6 | 92 | 92 | 23.8 | 23.8 |
| 3 | 28.0 | 102 | 102 | 28.0 | 28.1 |

The course par is 72 for the cases above

These test results confirmed that the application logic was coded correctly when calculating each players gross score. However, this was not the case initially because each calculated total did not allow for changes in the score data. The algorithm for this had to be amended so that every time a player scored over a double bogey (2 shots over par) on a hole their score needed to be reduced to this value. Following this the above results were produced, however there still exists a fault with test three. The system was intended to be implemented so that it only allowed handicaps up to 28.0 and as shown above an amended handicap has the value 28.1. A complete system would allow female players to obtain handicaps up to 36.0, however this wasn't the intention of this solution because it would require a greater amount of application logic, and so this couldn't be included especially with the project schedule being so strict. Therefore, in order to overcome this problem discovered in test three, the code was changed so that if a player has a handicap of 28.0 and they posted a score higher than this, then there would be no change to this value, because this is the maximum allowed.

## Net score calculation test

This comparison is similar to the previous test however the handicap is taken from the gross score in order to produce a net value. This value produced by the application is then compared against an expected score as well as an expected handicap.

| Test | Handicap | Expected net score | System net score | Expected handicap | System handicap |
|---|---|---|---|---|---|
| 1 | 12.9 | 63 | 63 | 10.2 | 10.2 |
| 2 | 26.6 | 65 | 65 | 23.8 | 23.8 |
| 3 | 28.0 | 74 | 74 | 28.0 | 28.0 |

The course par is 72 for the cases above

Following the amendments to the algorithm described above, all of the test results matched their expected values. This is because these scores are calculated from the gross values obtained and so this adjustment aided these calculations.

## Stableford score calculation test

The purpose of this test is to determine an expected number of total points from the test data and this number is then compared against the value returned from the system with the same data. The calculation uses both the par and stroke index of each hole in order to determine the number of points scored, these are then tallied to produce this total points value for comparison. Below are the points calculated for every hole for each of the tests.

| Hole | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Par | 4 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 5 | 4 |
| S.I | 1 | 15 | 5 | 6 | 17 | 13 | 8 | 12 | 7 | 16 | 2 | 18 | 14 | 3 | 7 | 11 | 10 | 4 |
| | | | | | | | | | | | | | | | | | | |
| Test | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 | P15 | P16 | P17 | P18 |
| 1 | 2 | 2 | 2 | 3 | 1 | 3 | 3 | 3 | 2 | 2 | 3 | 4 | 3 | 3 | 2 | 2 | 2 | 3 |
| 2 | 2 | 1 | 2 | 4 | 3 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 3 | 2 |
| 3 | 2 | 1 | 2 | 3 | 2 | 3 | 3 | 2 | 3 | 1 | 2 | 2 | 3 | 4 | 3 | 2 | 1 | 5 |

The points calculated from the table above have been summed and this actual value will be used for comparison with the results obtained from the application.

| Test | Handicap | Expected points score | System points score | Expected handicap | System handicap |
|---|---|---|---|---|---|
| 1 | 12.9 | 45 | 45 | 10.2 | 10.2 |
| 2 | 26.6 | 44 | 44 | 23.8 | 23.8 |
| 3 | 28.0 | 32 | 32 | 28.0 | 28.0 |

The course par is 72 for the cases above

In conjunction with the previous tests involving the net differential scores, these three also proved successful because all of the expected points results matched those calculated by the system. In addition to this, the handicaps were also calculated correctly.

Conclusions from testing competition results

As far as all of the above comparisons are concerned, most of them matched the values, which were expected. However, as previously mentioned there was is a slight problem with the algorithm used to calculate a member's new handicap following their participation in a competition. With the thorough testing of each of these methods, it became apparent that the actual algorithm requires the removal of outliers from the member's data (depending on their handicap) when they play an extremely bad hole. Furthermore, [8] explains that handicaps are adjusted by five different factors and are split into five categories with different buffer zones in each, where as only three of these were implemented. Therefore, these changes were implemented in the code so that there were no discrepancies with these calculations.

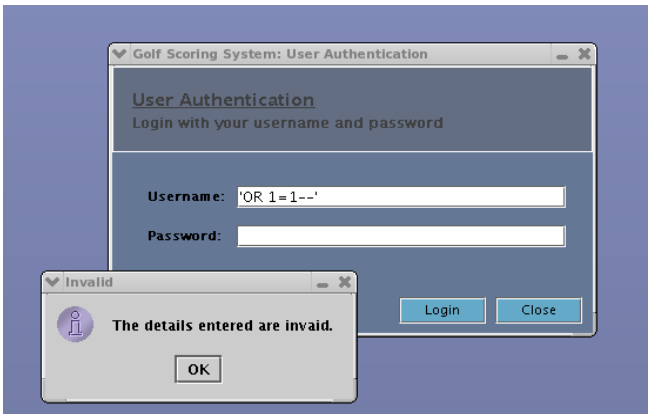
## 5.3.2 Login testing

In order to effectively test the login process, the user names and passwords that already exist in the database were tested against to find whether access was granted to the system. As this was successful another test was required to determine whether the system was vulnerable to an SQL injection attack. The system was tested against the following code:

- 'OR 1=1--'

The result of the test was that the system was able to combat against these attacks. As previously mentioned this was because the login method was implemented so that it used regular expressions where only specified input is accepted. This is shown in Figure 5.1.

Figure 5.1 Login validation




### 5.3.3 Competition constraint testing

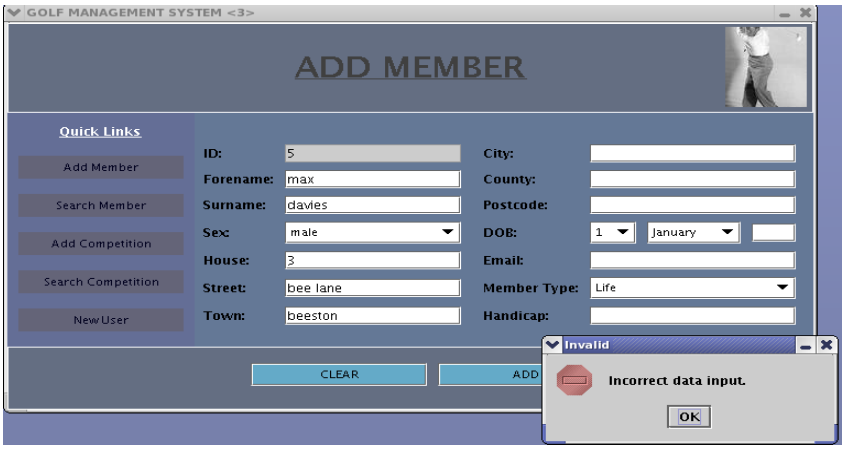
As previously stated in the design chapter, it is essential that the mechanism involved with adding member's scores only allows one set of results for each member in a competition. Therefore there is a need to test this part of the system to see whether the code implemented could recognize more than one set of results for a member. This method proved successful, because an error would appear in the case of this type of input (as shown in Figure 5.2).

Figure 5.2 Scorecard validation

## 5.3.4 Validation testing

A final aspect of testing is checking whether the validation coded during implementation is suitable to match the systems requirements. Each validation test will entail input into the system checking the input mask set for each interface. It's purpose is to stop unwanted data being stored in the database, and it hopefully achieves this by alerting users that their input is unsuitable in cases where input does not match the data requirements of each field on a form. Following all tests checking the validation for empty fields, maximum characters, invalid input it was found that all the input masks introduced during implementation worked effectively not allowing any invalid input. This is shown below using the add member interface in Figure 5.3.

Figure 5.3 Member validation




# Chapter 6: Evaluation

## 6.1 Introduction

Evaluation is the, “assessment of an information technology product or system against defined security-related functional and assurance criteria, performed by a combination of testing and analytic techniques” [3]. Therefore, in order to discover whether the application is a success or not, this chapter discusses functional, security and usability issues in detail. The knowledge obtained from these investigations can then be used when drawing conclusions about the system, towards the end of the chapter.

## 6.2 Evaluation against minimum requirements

To produce a thorough evaluation of the system in terms of whether it meets the original requirements, it is important to discuss each of the above points in turn. Information gathered from this section can then be used when drawing conclusions from this development. The following addresses issues such as implementation problems and successes as well as the outcome of each requirement.

### 6.2.1 A database containing golf members information

The system has been developed so that it is completely reliant upon the data held in the database. Therefore it currently runs where it is constantly opening and closing a database connection depending on

what process is being carried out. In meeting this requirement there were few stumbling blocks during development. As previously mentioned this phase required two iterations for completion because it was not realised that the stroke index field was required for the stableford calculations. Although this was the major flaw with the database implementation, there were also other smaller problems during this phase.

The production of an auto-incrementing primary key proved difficult because although a method was discovered within the Postgresql manual this failed to integrate with the database set-up. To overcome this problem, a simple count was made of the records currently in a database table and so each new record would be assigned an ID value of this plus one.

A less challenging problem occurred, when assigning a data type for the handicap field. It was decided that this required a floating-point value to one decimal place (as members handicaps are recorded in this way, e.g. 12.6) to achieve the correct data type. This task was overcome through thorough searching of the Postgresql manual to define a specific numeric type. Apart from these few challenges, implementing this section proved fairly trivial when comparing this task with the other requirements.

### 6.2.2 The system must allow data entry from scorecards on a user interface

This requirement proved one of the simplest to implement. However, it could not be addressed until all of the interfaces had been implemented and linked together. This is because the mechanism behind this, meant that a competition search was required, and this interface could then be accessed from the competition details interface as shown in Appendix F**.** Only one difficulty was found where the competition ID from the competition details interface, needed to be passed onto this class so that each set of results could be assigned to the correct competition. Following the completion of this, results were then validated (checking for empty fields and digits) and a constraint was introduced so that a member was only eligible to play competition once (as shown in Figure 5.2). Following these checks on the data provided, the task of inserting each set of results to a record in the database was simple to code.

Much like the last, this requirement also proved fairly simple to implement and is also a small part of the result calculation process. However, it is deemed as very important because without successful completion, the main functionality of the system could not be addressed.

### 6.2.3 Output must include a list of competition results after data processing

Out of all of the requirements set at the start of this project, processing the results entered from a competition proved by far the most difficult. As previously mentioned from conclusions in the previous chapter, testing the handicap changes showed some minor errors with calculations. This process proved to be very time consuming because it dealt with calculating results for three different types of competitions as well as the handicap changes involved in each. When breaking these tasks down during development both the gross and net differential calculations were far more easily implemented than the Stableford.

The procedure involved adjusting scores inputted so that they were not outstanding (scores needed to be changed to a double bogey if they were higher than this) this data was then stored in the scores table in the database. Following this, these values were then retrieved and processed accordingly depending on which type of competition they were for. As previously stated, the stableford calculations were reliant upon comparisons of each hole value, with both the stroke index and par of the specified hole. In addition to this, these comparisons were then turned into points values, which needed to be summed for the overall competition results.

It seems there are both positive and negative arguments involved with the fulfillment of this requirement. Although this process eventually turned out to be a success (because all of the results were calculated accordingly following certain code modifications), it did however turn out to be far more time consuming than intended and so this is the reason why few of the further requirements could be fulfilled. Furthermore, the calculations involved are still not completely accurate because they do not make use of any standard scratch score (SSS) and instead use the course par for all their comparisons. With this in

mind, the system fails to strictly abide by the CONGU® Unified Handicapping Regulations of 2004. However, this is something that cannot be calculated when testing this existing system because it depends on the state of the course on a particular day, which affects each member's scores. Taking this into account it would not be reasonable to integrate this aspect of golf scoring into the competition processing, and although the results contain this minor inaccuracy, they are however, extremely reliable in their calculation without this.

### 6.2.4 Must have the ability to deal with changes to the database through a User interface

This is another requirement which took far longer to complete than expected. It involved two phases, which included both, update and delete functions. In order to help achieve consistency throughout the application it was necessary to implement both of these for members and competitions. These tasks did not prove too difficult, even though there were some extra features incorporated into this requirement.

As a direct manipulation system, this procedure involves a member's record being searched and then selected. Following this, the user has an option to modify the details displayed or to delete that member from the database. An additional feature incorporated into this, is that the initial details displayed are not editable, so if the user wishes to modify these, they must select the option available so that the necessary fields become editable. Although this feature was deemed to be a necessary part of this procedure, it was not stated in this requirement and so it has been realized that only concentrating on aspects for this requirement would have allowed more time on implementation with advanced requirements. Despite this, during implementation it was decided that providing this editing feature was imperative as it helps avoid the possibility of human error, in cases where users edit profiles accidentally.

## 6.3 Evaluation by Monmouth Golf Club

A major part of the evaluation process is allowing other golf clubs to test this solution themselves because, "evaluation is relied on quite heavily by HCI" [4]. In order to test the system in this way, contact

was made via email with the secretary of Monmouth Golf Club (MGC), Peter Tulley. As a previous member of this club, Mr.Tulley welcomed the prospect of evaluating this application. Therefore, I was invited to the golf club on 26th March 2007 to present my project.

### 6.3.1 Preparing for the presentation

In order to do this, it was soon realized that the applications files needed to be transferred onto a CD so that it could be used externally. This task was done using a build file so that all the code could be zipped then added to a CD. Once this was inserted into an optical drive on a windows SOC machine, the program installed and ran successfully (as shown from the test in 5.2.1). This test proved essential for this evaluation phase so that I knew the system would run effectively on the workstation used by MGC. In the build up to the presentation I also compiled a list of questions for Mr.Tulley to answer after his evaluation. These can be found in Appendix D.

### 6.3.2 The presentation

During the presentation, I explained to Mr.Tulley what the purpose of this application was and the additional features it provides in comparison with existing solutions before demonstrating the software. This explanation was provided because MGC currently uses a Club 2000 system to manage their handicaps, and I wanted to make Mr.Tulley aware that the purpose of this solution was not only to match all of the functionality of their existing system, but also to further this with a few extra features. I managed to successfully install and run the program at which point Mr.Tulley was presented with the login facility. During the demonstration I managed to guide him through every aspect of the system finishing with the star members feature (which many existing systems don't contain). Following this I asked Mr.Tulley numerous questions (found in appendix D) about his experience with this system and gained some interesting feedback from this evaluation.

### 6.3.3 Presentation feedback

The main part of evaluating the software in this way is to gather effective feedback, which can be used to discover the success of this system. There must be a requirement for this type of system if it is to be successful and so it is often the case that just producing a working system is not enough in software development, as the solution must provide a purpose. Using Mr.Tulley's responses the process of determining whether there is scope for this type of application is made easier.

From the experience Mr.Tulley gained using this software, he was extremely pleased with the design of the interfaces as they provided, "excellent navigation through the system because you always knew where you were." It is also worth mentioning that Mr.Tulley was fond of the colour schemes chosen for the interface because he found that they "reflected the purpose of the system." A question was asked about the level of learnability provided by the system, to which he mentioned the system would be quite easy to learn. However, he also made the point that the software currently used by the golf club functions in a very similar way so this probably helped this level of learnability.

As far as the star members feature was concerned Mr.Tulley was pleased to see the inclusion of this because he had never seen anything like this before. A further question involving this feature asked whether it would be put to use especially with regards to awards ceremonies. His reply stated, "I think this could probably be put to use during the awards, however I think this would interest the members more, knowing that they could be the top golfers at this club. This could also be the case for the eclectic score which I feel is more interesting however, a disadvantage of this could be that many members have the same score and so there will be no lead player."

Another question put to Mr.Tulley asked for any advantages or disadvantages of this solution over the system currently used. As previously mentioned, at the start of the presentation I explained the main features of the system and made him aware that result calculations failed to use the SSS and instead compared each score to the course par. So Mr.Tulley explained that although this functionality wasn't

completely accurate, “it would have been nice to have the option to enter the standard scratch score somewhere when entering the competition results because this is what Club 2000 allows me to do.” In addition to this he also didn’t approve that the system failed to allow for competitions from either the red, yellow and white tees (this alters the course par in many cases, making it easier of red tees and harder off the white tees) and instead only took one of these into account. He made it clear that the Club 2000 system currently used, allows for each of these and so the system would need to be adjusted to allow for at least three different course information inputs (whereas only one type is currently permitted).

However, despite all of these disadvantages Mr.Tulley stated, “if the functionality involving score calculations is sorted out, the system provides a cutting edge on Club 2000 because of the star members feature which it provides.” He was also surprised to see the personalization used on the scorecard where the hole names appeared as labels and mentioned that this was another feature, which he had not experienced before.

A final question posed to the user for this evaluation prompted him to suggest any possible improvements for the system. Where these are concerned Mr.Tulley expressed that improvements need to be made with the calculation as discussed above. Additionally, he mentioned that no improvements needed to be made to the outlook of the system because, “this competes with the current software we use.” However, he did point out that the scorecard interface should represent an actual scorecard in the sense that it should allow tabbing vertically rather than horizontally, as this can be confusing for the user. A final possible improvement was that the system should allow for the modification of user details rather than deleting a user and then having to make a new one, which is the mechanism currently used for this process.

## 6.4 Feedback from users

“Usability is typically measured by having a number of test users use the system to perform a pre-specified set of tasks, though it can usually be measured by having real users in the field perform whatever tasks they are doing anyway” [16]. Taking this into account, the system has already been tested on an

ideal user (Mr.Tulley at MGC), therefore in order to gain further knowledge on the systems level of usability, it is perhaps necessary to allow general users to evaluate the system as well.

### 6.4.1 General users

Using a few students in the SOC's Dec-10 laboratory, I compiled a short list of tasks (available in appendix D) for them to complete and following this I managed to gain some feedback by asking similar questions to those mentioned in my presentation at MGC. Although these users had no existing system to compare against and none had any real knowledge of this type of system, many commented on the user interfaces being attractive and thought they were well suited to the purpose of the application. Having little knowledge with the golfing environment none of them picked up on the fact that calculations were slightly inaccurate with the exclusion of SSS. However, many were pleased with the navigation and made use of the quick links panel provided. Despite this, there were a few users which prefer the traditional bread crumb trail technique for this navigation so are fully aware of how they can return to the previous screen. As a result of this, they would have preferred this process rather than use the quick links and the main menu for their navigation.

## 6.5 Future project extensions

The knowledge gathered from all these different evaluation techniques has been instrumental in deciding what can be done to further this project. With the entire list of minimum requirements being met, it has been decided that the following features could all be integrated with the existing solution, thus furthering its capabilities. With time permitting, some of these would have definitely been implemented to achieve this, however the existing schedule was fairly demanding when trying to fulfill the initial requirements and so the list below could not be addressed.

### 6.5.1 SSS inclusion

This feature must be regarded as the most important because it is essential not to have any inaccuracies in the calculation of competition results and changing handicaps. This could be included, by requiring some

input from the user when they are about to enter a set of results. Each result tally must then be compared with this value rather than the course par, which is currently being used. Following the integration of this, all calculations would be completely accurate with both the results they produce as well as the adjusted handicap for each player.

### 6.5.2 Member's best hole facility

It is the intention that this facility will be integrated with the star members feature in the existing system. This idea is based on finding out which members post the lowest scores on each hole. As a previous member of MGC I would be interested in this and I am sure many other golfers would be as well. It carries much the same purpose as the eclectic score facility where members would be interested to find out whether they are the best performers on a certain hole at their course.

### 6.5.3 Lesson booking system

Incorporating this feature into the system would provide a mechanism for all types of users. Therefore while the club secretary concentrates on managing the members and competitions, the club professional would be able to book lessons with members on a calendar. It is intended when using this feature that these two types of users will only be allowed to gain access to parts of the system, which are solely implemented for them. The logic behind this lesson booking system is that it will provide a more complete golfing solution because it provides functionality for the main personnel at a club.

## 6.6 Evaluation of project management

The most important part of systems development is trying to stick to the schedule. It is always difficult to overcome milestones knowing that every part of the system is intended to function properly by a set date. The following discusses whether this schedule was strictly followed throughout the background research, analysis, design, implementation and testing phases of this project. Additionally, this knowledge will be used help determine whether the schedule was realistic or not for this type of development.

### 6.6.1 Did I manage to stick to the schedule?

As far as the earlier stages of the projects development are concerned, it was fairly easy to stick to the schedule during the background research and analysis studies. However, it was not until the design phase that I started to fall slightly behind schedule. As previously noted in the design chapter there were a few problems when trying to implement the database design. The unified process was integrated with the waterfall model so that it can be quickly repeated over a number of iterations. Therefore, this meant that once these errors were discovered with the database, the design had to be changed and implemented resulting in a further iteration. It was deemed essential to implement this correctly because this acts as the platform for the entire application.

As this put the rest of the implementation process slightly behind schedule, it was imperative that the next few phases of development ran smoothly so that this initial problem could be overcome. This proved to be the case as the design and implementation of the user interfaces was completed before expected. The task of coding the back end classes, which are utilized by each user interface, was the next stage of development. Designing and coding the competition results was by far the biggest task within this section. This caused various problems when designing and coding and this is why it required three iterations for completion.

Following this setback, it was decided that only one of the minimum requirements could be completed in time for the presentation at MGC because the remaining time was allocated for system testing. Therefore, the most unique feature was chosen out of the advanced requirements list, and so the star members function was designed and implemented along with an eclectic score facility. Although this proved a simple task it was still quite time consuming because a user interface needed to be designed and implemented to show this data. As previously mentioned, the remaining time was used for system testing. The time allowed for this was minimal, however testing was carried out on all the main aspects of the system, which is demonstrated in the previous chapter.

### 6.6.2 Was the schedule realistic?

The discussions above show that development didn't always go as planned and therefore the schedule was not strictly followed. A reason for this is that the size of a few tasks were not estimated effectively (for example calculating a new handicap from competition data) and so the total time taken to code these were not taken into account. However, there were a few noticeable stumbling blocks when calculating competition results and producing an eclectic scoring feature. These problems during system development could not be foreseen and so it could be argued that the schedule was entirely suitable for this project.

The methodology adopted proved successful because each section of implementation was coded until it functioned correctly using a number of iterations. Therefore, it has been assumed that without following this, the project would have been further behind schedule than it turned out. This is because a methodology could have been followed which allowed system errors to build up until the end of development, which would have been far trickier to fix.

Overall, I think that the schedule was slightly unrealistic however this argument is based on the number of problems encountered during development. As previously mentioned, although testing was minimal, it still proved successful because the output was a complete working system which was the intention from the beginning. Initially, it was essential that all of the minimum requirements were fulfilled and that the solution was fully tested. As far as the project deliverables are concerned, everything has been fulfilled except from the production of an administration manual. The Gantt chart in Appendix B shows that the time remaining from the end of March until April 24$^{th}$ was used to work on this project report and produce a user manual. It was therefore decided that producing an administration manual would not have been an efficient use of the time remaining especially when i am the only current administrator. With this in mind, such a manual was deemed to be fairly trivial. Although the project management of these tasks wasn't carried out as intended, the solution still provides all of the necessary functionality including some advanced features, so the project should be deemed a success as a result of this.

# Bibliography

[1] Avison, David. Fitzgerald, Guy. Information Systems Development: methodologies, techniques and tools (4th edition). Maidenhead:McGraw-Hill Education, 2006.

[2] Club Systems International Limited. URL:http://www.club2000.co.uk/. [10th October 2006]

[3] Evaluation definition (2001). URL:http://www.atis.org/tg2k/_evaluation.html. [18th March 2007]

[4] Faulkner, Christine. The essence of human-computer interaction (1998).

[5] Get Physical Software. URL:http://www.ClubManagementSoftware.com. [14th October 2006]

[6] Graphics design versus usability. URL:http://www.experienceddynamics.blogs.com/site_search_usability/would_you_like.html. [8th December 2006]

[7] Golf rules in brief (1997). URL:http://www.golfeurope.com/almanac/golf_rules.htm. [18th September 2006]

[8] Handicap calculation (2004). URL:http://www.mygolfexperience.com/golf/uk/golf-handicap.asp. [26th January 2007]

[9] Handicap Master Limited (1998). URL:http://www.handicapmaster.org/. [12th October 2006]

[10] How to calculate your handicap in golf (1999). URL: http://www.ehow.com/how_15107_calculate-golf-handicap.html. [28th January 2007]

[11] Hunt, John. The Unified Process for practitioners, object oriented design, UML and java. Practitioner series, 2001.

[12] Jencess Software and Technologies (2002).
URL:http://www.jencess.com/products/connectsuite.html. [12th October 2006]

[13] Johnson, Owen. SE24: The Unified Process notes. University of Leeds, 2006.

[14] Making the right colour choices (2004). URL:http://www.builder.com.com/5100-31-5073173.html. [15th December 2006]

[15] Manino, Michael V. Database design, application development and administration (3rd edition). Maidenhead:McGraw-Hill Education, c2007.

[16] Nielsen, Jakob. Usability Engineering (1993).

[17] Pinedo, Michael. Scheduling – Theory, algorithms and systems (2nd edition, 2002)

[18] Savitch, Walter. Absolute C++ (2nd edition). London:Addison-Wesley, c2006.

[19] Shneiderman, Ben. Designing the user interface strategies for effective human-computer interaction (3rd edition 1992). University of Maryland.

[20] Skelton, John. Bennett, Simon. Lunn, Ken. UML (2nd edition, 2004). Schaums outlines.

[21] Usability versus navigation (2000). URL:http://www.jessett.com. [12th December 2006]

[22] Why I teach eclipse? (2005). URL:http://www-128.ibm.com/developerworks/rational/library/jun05/pollice/. [5th September 2006]

[23] Wilson, Brian. Systems: concepts, methodologies and applications (2nd edition). Chichester: Wiley, c1990.

[24] Wilson, James M. Gantt charts: A centenary appreciation. Management studies department at Glasgow University.

[25] Wu, C.Thomas. An introduction to object -oriented programming with Java (4th edition). New York:McGraw-Hill Education, c2006.

# Appendix A

It is essential when choosing a project, to choose something of interest and in this case a system built around golf was the perfect challenge for me. As a golfer myself, I have always had a rough idea about how competition calculations work however, this project has taught me this exact process. Not only does this further my knowledge of the game itself but in producing this solution I feel that my knowledge of software development has grown throughout. In taking on this project I have had to use all aspects of systems development, which I have either been taught or researched in order to overcome obstacles during this lengthy process.

My skills involving time and project management have been furthered because the schedule produced at the beginning proved very challenging. I have also had to deal with pressure situations like completing coding sections to meet set milestones throughout development. It has also been realised that making sensible judgements of how to allocate tasks is of paramount importance because it is these decisions, which provide the scope of the system. This is determining what can be done in the time allocated. It is often hard to know when to stop coding with a project like this because there can be scope for so much more functionality. Therefore, it was deemed impossible for every advanced requirement to be complete in time especially when time for error fixing and testing cannot really be predicted to a useful accuracy. This proved a major flaw in development and it has now been realised that it is not very often the case that everything runs smoothly no matter how influential the project management is.

I fell slightly behind schedule with development because errors needed to be eradicated from the solution. However on many occasions when overcoming these challenges I found that I improved both my programming and problem solving skills. This is all part of the experience gained from developing a system and I think that I have gained far more out of this than I would have if everything I designed got implemented with ease. I feel that is not the purpose of taking on a project like this, it is intended to be a

challenge and so when things don't go right, correcting some functionality under pressure is where I gained my most useful experiences.

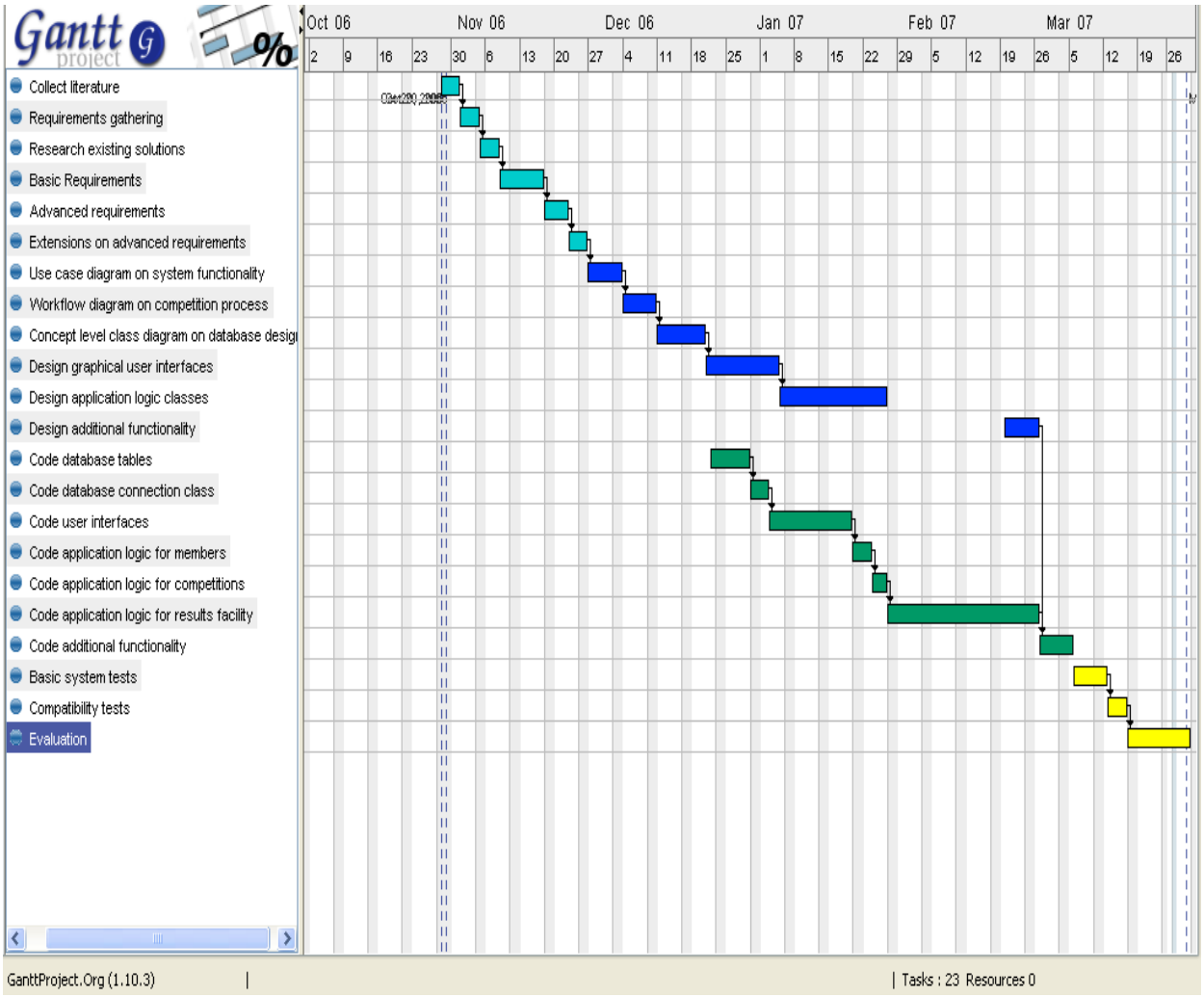
In particular I feel that my programming skills have vastly improved from what they were at the start of this project. In conjunction with this, I have realised the benefits of Java over neighbouring languages such as C++ and this knowledge will help me with other projects I take on. In addition to this I have researched other text editors available like IDEA, which I didn't know existed. It has been discovered that it is vitally important to choose software tools and the language you are going to use very carefully because these must provide the developer with appropriate support for the problem at hand.

Although the evaluation process gave me some excellent tips on how to modify the system, I felt that because I knew Mr.Tulley from being a member at MGC the presentation experience wasn't as professional as I would have liked. Although his evaluation of the system was not influenced by this (which was evident in the criticisms he gave) I think that for future reference a better experience would have been gained if I presented my application for some other golf clubs.

I am definitely interested in furthering this project because I feel that so much more can be done in terms of integrating new functionality with the existing solution. The additional requirements not covered would be a good starting point for anyone wishing to improve this application, so that it becomes a more complete golfing solution, with features covering all aspects of managing a golf club.

# Appendix B

## GANTT CHART FOR THE SCHEDULE




The time remaining from the end of March until the 24th April was used to produce a user manual and to improve the project report. For further information on Gantt charts visit [24]. For additional information on scheduling see [17].

# Appendix C

## Glossary Of Golfing Terminology

CONGU - *A CONGU Handicap is recognised by all Golfing Unions and Associations throughout the world as indicative of the standard of play of the player presenting it.*

Gross Competition – A competition which is based on the number of strokes a golfer plays for their round. The lowest score will win the competition.

Gross Score – *This value is the score a golfer obtains on each hole they play during a round. The total gross score is the sum of all of these.*

Handicap – *A numerical measure of an amateur golfer's playing ability. It can be used to calculate a net score from the number of strokes actually played, thus allowing players of different proficiency to play against each other on equal terms.*

Net Competition – *A competition that is based on tallying each players net differential scores on every hole. The lowest score will win the competition.*

Net Differential Score – *This value is the score a player gets on a hole taking into account their handicap and the number of shots they get. The total NDS is the sum of all of these.*

Par – *A predetermined number of strokes that a golfer should require to complete a hole, a round (the sum of the total pars of the played holes), or a tournament (the sum of the total pars of each round). They are the central component of stroke play, the most common kind of play in professional golf tournaments.*

Stableford Competition - *A competition that is based on tallying each players Stableford points obtained on every hole. The highest score will win the competition.*

Stableford Score - *This value is the number of points a player gets on a hole taking into account their handicap and the number of shots they get. The total Stableford score is the sum of all of these points.*

Standard Scratch Score – *This value is the score a player off scratch is expected to shoot depending on the course conditions.*

# Appendix D

## Questions posed during presentations

1. Does the system functionality compare with club 2000?

2. Do the user interfaces provide an effective outlook for this application?

3. Does the navigation provide a good level of system learnability?

4. Have you used a star members feature before? If not what do you think it would be of use?

5. What are the relative advantages and disadvantages of this system compared with club 2000?

6. Any possible improvements?

## Tasks set for presentations

1. Attempt to login to the system.

2. Add a member from the main menu interface.

3. Search for that member you have just added.

4. Modify their details and update their record.

5. Add some competition details.

6. Search for that competition.

7. From the competition details interface, add some scores providing the member ID for the member created in 2.

8. View the results for this competition.

9. Attempt to close the system.

# Appendix E

# GolfDbSchema.sql

```
drop table members cascade;
drop table competitions cascade;
drop table scores cascade;
drop table users cascade;
drop table holeInfo cascade;
drop table courseInfo cascade;

create table members(memberID integer primary key,
                    forename varchar(12),
                    surname varchar(12),
                    sex varchar(6),
                    houseNameNo varchar(15),
                    street varchar(20),
                    town varchar(15),
                    city varchar(15),
                    county varchar(15),
                    postcode varchar(7),
                    DOB varchar(10),
                    email varchar(20),
                    memberType varchar(25),
                    handicap numeric(3,1),
                    handicapChange numeric (3,1));
```

```
create table competitions(compID integer primary key,
                compName varchar(15),
                compDate varchar(10),
                compType varchar(20));

create table scores(compID integer REFERENCES competitions,
                memberID integer REFERENCES members,
                hole1 integer,
                hole2 integer,
                hole3 integer,
                hole4 integer,
                hole5 integer,
                hole6 integer,
                hole7 integer,
                hole8 integer,
                hole9 integer,
                hole10 integer,
                hole11 integer,
                hole12 integer,
                hole13 integer,
                hole14 integer,
                hole15 integer,
                hole16 integer,
                hole17 integer,
                hole18 integer,
                total integer);
```

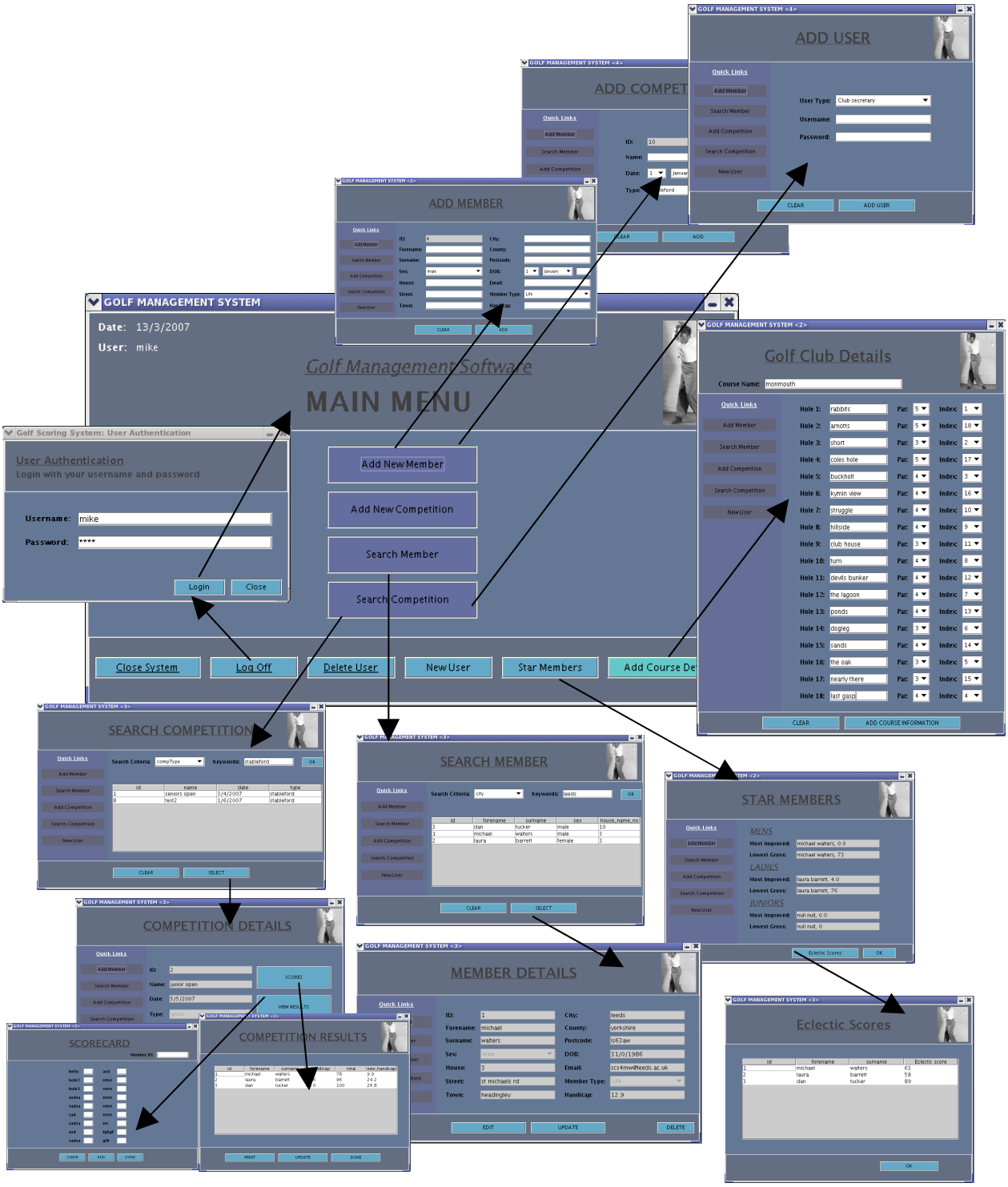
```
create table users (username varchar(15) primary key,
                    password varchar(10),
                    userType varchar(20));

create table holeInfo(holeNumber integer,
                    holename varchar(15),
                    strokeIndex integer primary key,
                    par integer);

create table courseInfo(courseName varchar(20) primary key,
                    coursePar integer);
```

# Appendix F

## User Interface work flow



# Appendix G

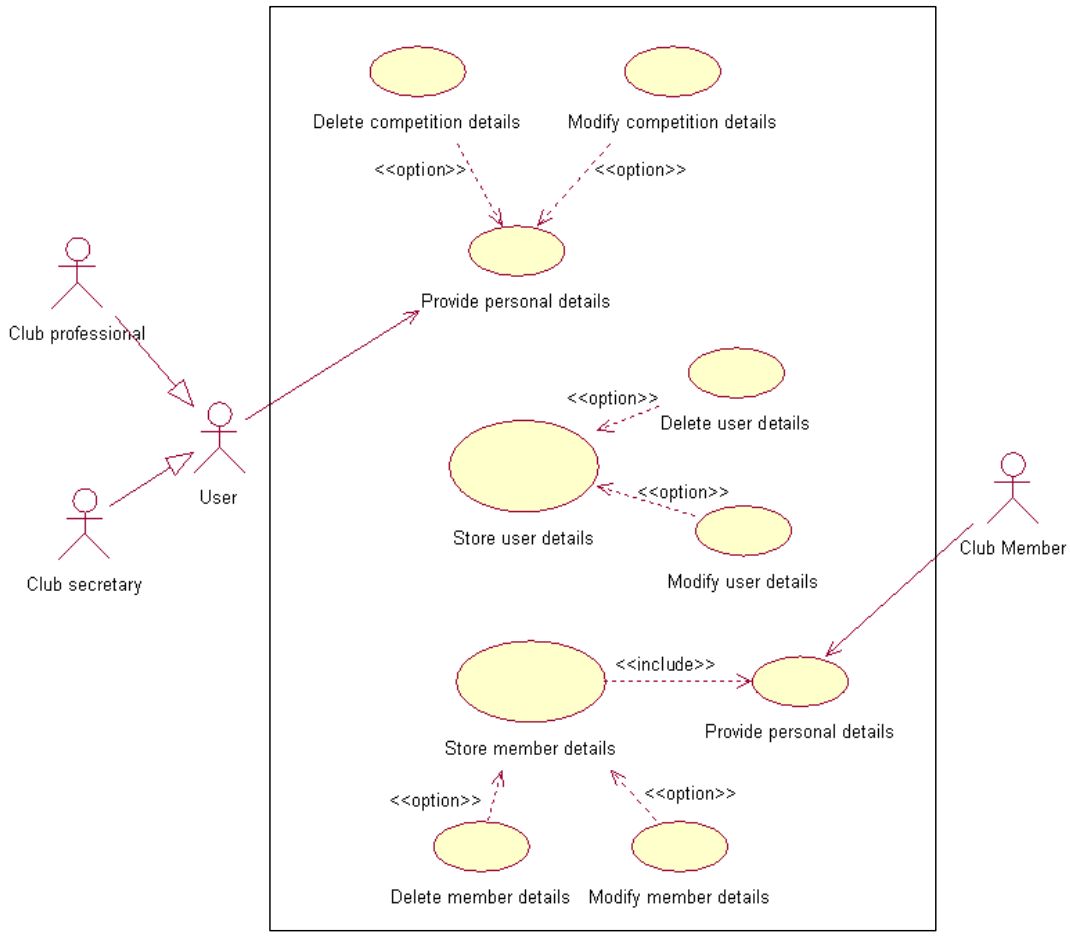
## Stableford Points Calculations

```
public int calculatePoints(int scoreX, int strokeX, int parX, float handicap) {
    int net = 0;
    int points = 0;
        if (handicap > 18.0) {
                if (strokeX > 10) {
                        net = scoreX-2;
                }
                else
                        net = scoreX-1;
        }
        else if (handicap <= 18.0) {
                if (handicap >= strokeX) {
                        net = scoreX-1;
                }
                else
                        net = scoreX;
        }
        int score1 = parX-4;
        int score2 = parX-3;
        int score3 = parX-2;
        int score4 = parX-1;
        int score5 = parX;
        int score6 = parX+1;
        if(net == score1) {
                points = 6;
```

```
        }
        else if(net == score2) {
                points = 5;
        }
        else if(net == score3) {
                points = 4;
        }
        else if(net == score4) {
                points = 3;
        }
        else if(net == score5) {
                points = 2;
        }
        else if(net == score6) {
                points = 1;
        }
        else {
                points = 0;
        }
        return points;
}
```

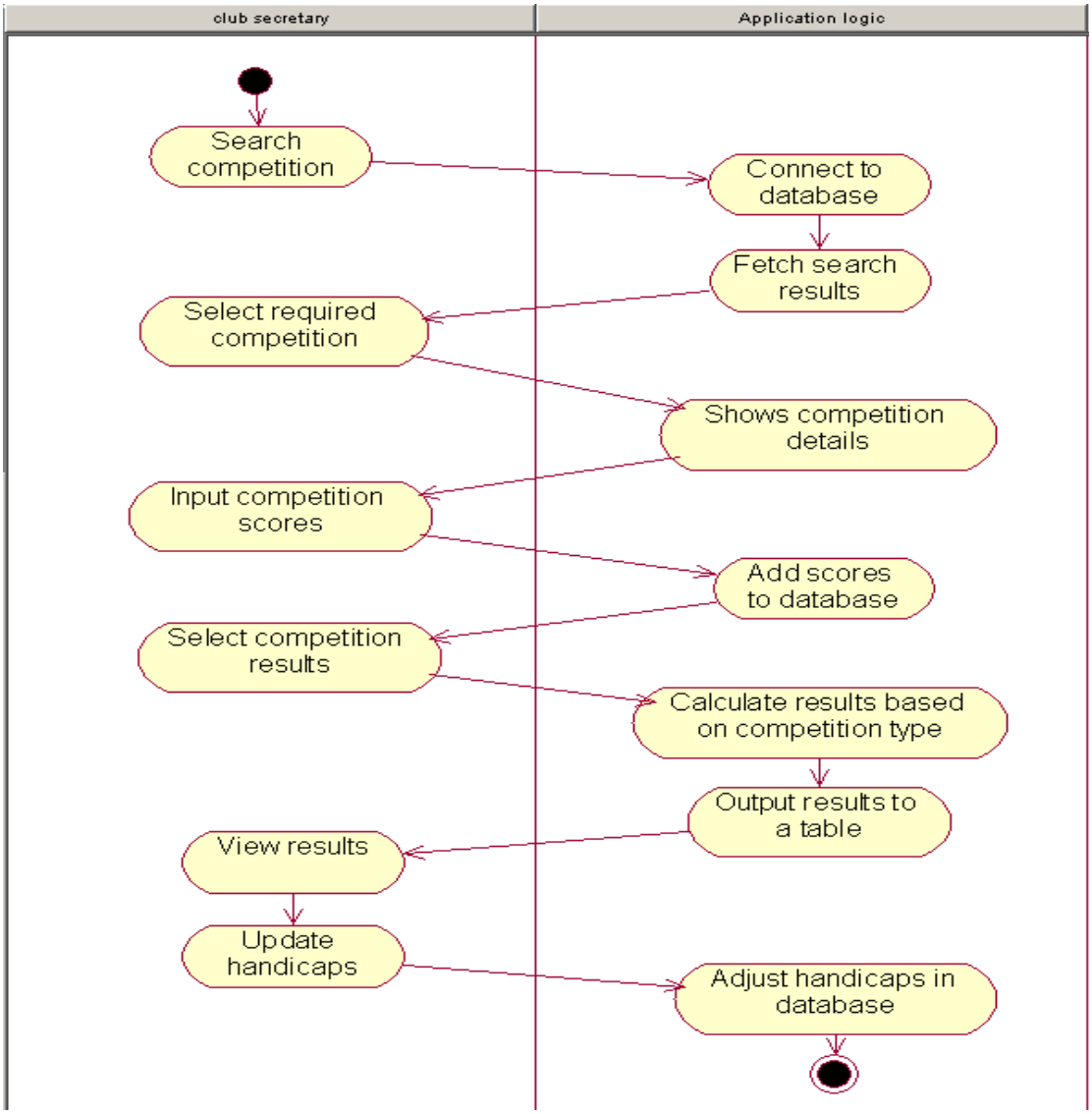
# Appendix H

## Use case diagram for registration process



# Appendix I

## Work flow diagram of competition calculations



# Appendix J

## Nielsen's Usability Principles

The following quotations are from [16].

Visibility of system status

“The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.” For example, in the desired system the secretary should be alerted once they are logged in correctly.

Match between system and the real world

“The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms.” Therefore the desired system should provide golfing terms so that the secretary or the professional are familiar with the language used.

User control and freedom

“Users often choose system functions by mistake and will need a clearly marked ‘emergency exit’ to leave the unwanted state without having to go through an extended dialogue.” In order to achieve this it is the intention that the system will strive to achieve an interface, which provides excellent navigational features.

Consistency and standards

“Users should not have to wonder whether different words, situations, or actions mean the same thing.” Therefore, the desired solution will aim to clarify any assets like these.

Error prevention

“Even better than good error messages is a careful design, which prevents a problem from occurring in the first place.” The design will incorporate every aspect of functionality for a complete golfing system and in using this will overcome any issues involving this.

Recognition rather than recall

“Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.” The interface itself will achieve this by making every piece of information valuable to the task at hand so that there will be no confusion from the users point of view.

Flexibility and efficiency of use

“Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.” As previously stated the system will provide excellent navigation so that users are completely aware of their location.

Aesthetic and minimalist design

“Dialogues should not contain information, which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.” Everything in the user interface will be of relevance and the developer will strive to achieve a maximum level of user visibility within this.

Help users recognize, diagnose, and recover from errors

"Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution." No codes will be used as error messages in this golf scoring system as the novice user will not be able to correct any errors. In addition to this, error messages will aim to constructively help the user to solve their problem situation.

Help and documentation

"Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation." Both novice and administrator documentation will be provided in order to help different kinds of users overcome any problems or to learn more about the application.

# Appendix K

## Proposed Graphical User Interface Layout

<table>
<tr><td colspan="2">The title of every interface will be here</td><td>The software logo will be here</td></tr>
<tr><td>The quick links panel will be here</td><td colspan="2">This is the main data section for every user interface within the application</td></tr>
<tr><td colspan="3">Any buttons will go in here</td></tr>
</table>

# Appendix L

# System Requirements

## Basic requirements

Using the knowledge gathered so far from the background research and the observations of existing solutions I feel that the following minimum requirements fully represent the main aspects problem domain. It is intended that this list of tasks will be developed first when producing my solution.

### A database containing golf members information

A database will be used to store details of each member and competition. In addition to this it will also be used to store scorecard data for each members round in a competition. In doing this further calculations can be done on this data to produce some useful information from the system like a list of competition results. It order to fulfil this task the database must also store some course details such as the par and stroke index of each hole so that the scores can be compared to these when producing results. Furthermore, the database will also store information involving users so this will aid system security as far as logging on securely is concerned.

### The system must allow data entry from scorecards on a user interface

The main function of this system is to receive data input in terms of each player's competition scores and to produce some output based on these. Therefore as each scorecard is being duplicated it is essential that these tasks are completed with a user interface which follows all principles of HCI described by [16].

### Output must include a list of competition results after data processing

As previously mentioned, the main functionality behind this system lies with the production of the competition results. It is the intention that once all of the scorecards from a competition have been inputted into the system that a list of results will be produced depending on the type of competition the members have played. It is imperative that this is taken into account because in most cases the competition details play a vital role in deciding the overall outcome.

### Must have the ability to deal with changes to the database through a user interface

This is an essential part of producing a system, which is consistent throughout. This is due to the intention of adding members, competitions, users and scores to the system. With the occurrence of this it must also be the case that each of these can also be either removed or modified, which means that these records must be amended in the database through a user interface.

## Advanced Requirements

After developing each of the requirements listed above, with time abiding the system could further its capabilities by integrating with some additional features. Using the knowledge gathered from implementing the basic requirements and also the background research completed beforehand I feel that the implementation of some of the following advanced requirements will create a competitive edge for the system with regards to existing solutions. This is because it is the intention that this Golf Management System not only has the capabilities of coping with scorecard data and managing members but can also be enhanced producing a more complete golfing solution.

### Integrating a club professional schedule into the system

This function will further the capabilities of the system because it is intended that members will be able to book lessons with their club professional using this calendar. The idea behind this is that members will visit their professional requiring a lesson and so the schedule will define the periods that the professional is available. This can then be used by the club professional to allocate an appropriate slot for the

member's lesson. This mechanism allows the club professional to schedule their activities in a effective manor knowing that the system will provide warnings where they make the mistake of double booking lessons.

## Incorporating member's fees into the system

It is the intention that only the club secretary uses this part of the system. It defines a method where the secretary has a record of who has paid their annual fees. Incorporating this into the system allows the user to overlook the current state of a member. An example of this is that they are able to see how long the member has left on their membership and so could send an email to this person offering a membership renewal. It is also the intention that this process should be automated and therefore with regards to this example, emails could be automatically generated and sent to members who only have a month left on their membership for example. This mechanism may also be used to chase members up if their fees are overdue.

## Introducing a points system for members

This provides a method where members get points for where they position in each competition they play. Therefore if a member has lots more points than others then they are playing extremely well. This system could therefore show the "star members" in a golf club and these personnel should be rewarded for their efforts. Not only does this improve a clubs relation with their members but it may also encourage members to better themselves as golfers having this in mind. This could then be put to use where annual awards are given to decide who has been the most consistent player, over a 12-month period.

An alternative approach to the point above is to maintain which players handicap changes by the largest margin over the year. This provides another helpful mechanism when determining who is most improved player for a whole season at a club.

## Introducing an eclectic scoring function

The logic behind introducing this into the system is that members would be able to see their best overall score after playing a number of competitions. The eclectic score is calculated so that it finds a members lowest score on each hole from all the competitions they have played in. It then tallies these values producing an overall lowest score for each member. This mechanism could be used to provide an extra niche to competitions with all members trying to better this score and it could therefore be included as part of an awards ceremony.

## Extensions On Advanced Requirements

Extensions on the above requirements provide an opportunity to further the system in a specific area depending on which of the advanced requirements have been undertaken. This development also creates a niche, which provides a "stand out" facility for the application. The following describes some of these opportunities and with time abiding some of these will be chosen for development.

### Allow members to access the system remotely

Incorporating this extension on the basic requirements would allow members to view weekly competition results and edit their own personal details for their golf club. This would mean that the secretary's role of constantly having to update member's details would be removed allowing them time to fulfill other tasks for their club. Furthermore, members would no longer be required to visit their club in order to find out where they placed in a competition. As a keen golfer, from personal experience I have often traveled to my local club in order to discover my place on a competition results list and would much rather be able to do this from my own PC at home. From researching these types of systems I realized that a few golf clubs are actually using this web based system so that their members are able to carry out tasks as previously described. However, there was no evidence that this has been incorporated into a clubs golf management software, which is intended of this extension.

### Incorporate an annual calendar detailing any social event that occurs at the club

This extension is fairly similar to the lesson-booking schedule for use by the club professional previously described. However, it is desired that only the club secretary will use this social calendar as their everyday activities involve organizing events for their golf club. It will work in much the same way as the previous schedule as it will provide warnings to the user where they are attempting to double book an event. As competitions will be scheduled with the Golf Management System it is not necessary to include them in this social calendar. The purpose of this is that it only schedules clubhouse bookings such as social events like a dinner and dance or karaoke night.

## Introduction of a desktop for the Golf Management System

This extension provides a mechanism, which aims to improve the interaction between the application and the users. Providing a desktop panel, which acts as a platform for the graphical user interface will create a more convincing outlook that Golf Management Software is intended as a piece of software for any golf club which requires a management facility. From studying many software applications it is apparent that those, which are most successful with the general public, are those that include a desktop. Therefore, integrating this with the desired solution will not only provide users with a feature which they commonly use but it will also follow many HCI principles described by [16] and thus improve the overall look, feel and interaction issues with the system.

# User Manual

**Michael Walters**

**Hardware requirements**

The following represents hardware requirements for the application.

INTEL CELERON D 356 Processor (or equivalent)

3.2MHz, 533MHz, 512MB cache

256MB DDR RAM

CD ROM drive

80GB hard disk

15” TFT monitor

**Operating System requirements**

The application runs on each of these operating systems:

Microsoft Windows 2000

Microsoft Windows XP

Fedora Core 6

**Installing the software**

Insert the CD into the optical drive of your PC

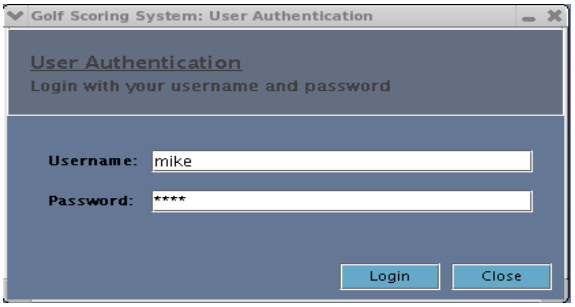
Copy the files into your main working directory

Double click on the executable file

**Logging on**

The first time you login to the system, your initial user name and password details will be recorded as the first user of the system. Enter these details into the text fields available and select the OK button. Logging on to the system in future (following this initial access), will require either these details or others (for new users). Figure 1.0 below demonstrates this process.

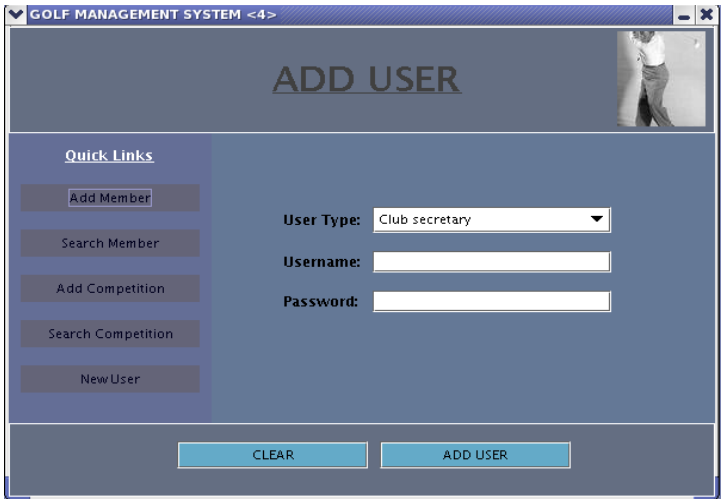
Figure 1.0 Login screen




**Adding a user**

In order to add a new user to the system it is essential to select this button from the main menu. Figure 1.1 below shows that this process requires a username, password and a member type. The criterion for the type of the member is either club secretary or club professional depending on the user. Once these details have been entered, you are required to select the add user button corresponding to this function. However, if you wish to change this input it is essential that you select the clear button and modify these details before adding this user. To return to the main menu simply select the cross situated in the top right hand corner of the screen.

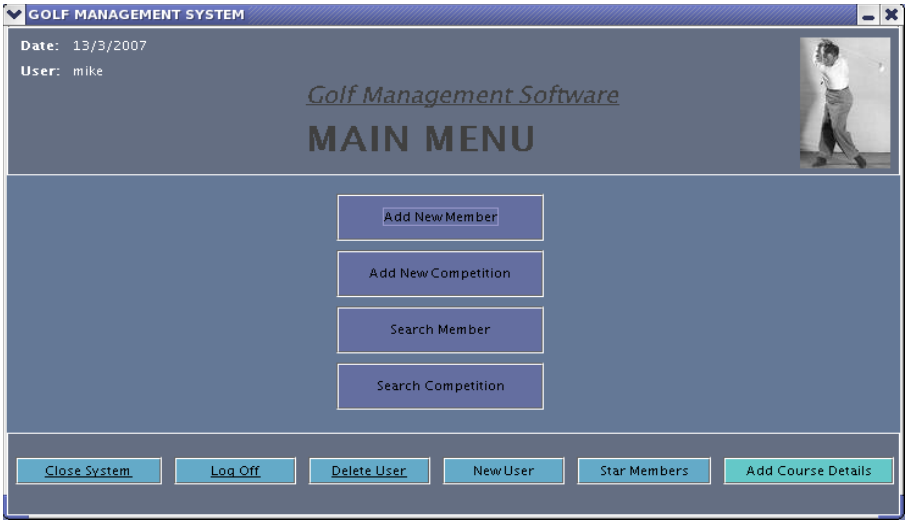
Figure 1.1Add user screen




**Deleting a user**

If you wish to remove a user currently logged on, simply select this button from the main menu shown below. Following this, you will automatically be logged out and the deleted users details can no longer access the system.

Figure 1.2 Main menu



**Adding course details**

Access to this interface can be obtained through the main menu. As shown below in figure 1.3 this requires a lot of user input. However, data such as the hole names are used to personalise this application (discussed later with score input). You are required to input the name of your golf club as well as information for each hole. This involves not only the entry of the par and name for each hole but also each stroke index. Following this, you must select the add course details button situated towards the bottom of the interface. Details of each hole have now been stored in your database. Clearing any inputted data can be done by selecting the appropriate button. As previously mentioned return to the main menu by selecting the cross in the right hand corner of the interface.

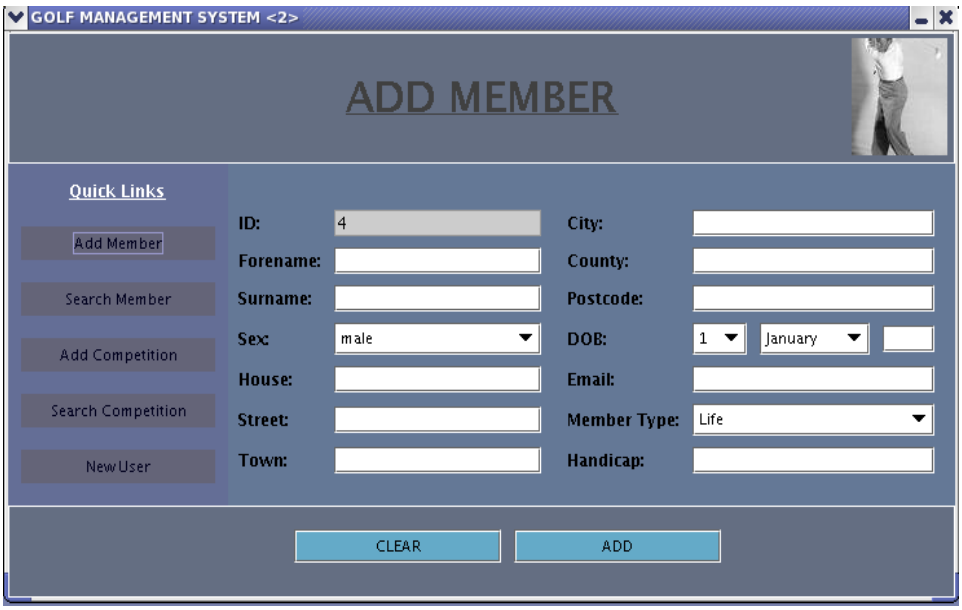
Figure 1.3 Add course details

GOLF MANAGEMENT SYSTEM <2>
Golf Club Details
Course Name: monmouth
Quick Links
Add Member
Search Member
Add Competition
Search Competition
New User
Hole 1: rabbits Par: 5 Index: 1
Hole 2: arnotts Par: 5 Index: 18
Hole 3: short Par: 3 Index: 2
Hole 4: coles hole Par: 5 Index: 17
Hole 5: buckholt Par: 4 Index: 3
Hole 6: kymin view Par: 4 Index: 16
Hole 7: struggle Par: 4 Index: 10
Hole 8: hillside Par: 4 Index: 9
Hole 9: club house Par: 3 Index: 11
Hole 10: turn Par: 4 Index: 8
Hole 11: devils bunker Par: 4 Index: 12
Hole 12: the lagoon Par: 4 Index: 7
Hole 13: ponds Par: 4 Index: 13
Hole 14: dogleg Par: 3 Index: 6
Hole 15: sands Par: 4 Index: 14
Hole 16: the oak Par: 3 Index: 5
Hole 17: nearly there Par: 3 Index: 15
Hole 18: last gasp Par: 4 Index: 4
CLEAR
ADD COURSE INFORMATION

**Adding a member**

To carry out the task of adding member's details to your database, simply select this button from the main menu. Following this, the interface will appear as shown in figure 1.4. Data input is required involving many personal details for that member so the club maintains a record of these. In addition to this you are also required to enter this member's handicap and member type. You will notice that each member added to the system will be automatically assigned a member ID. Any inputted data can be cleared by selecting the button corresponding to this. Once you are satisfied with your input, select the button to add this member to your database. As previously stated return to the main menu by selecting the cross.

Figure 1.4 Add member details

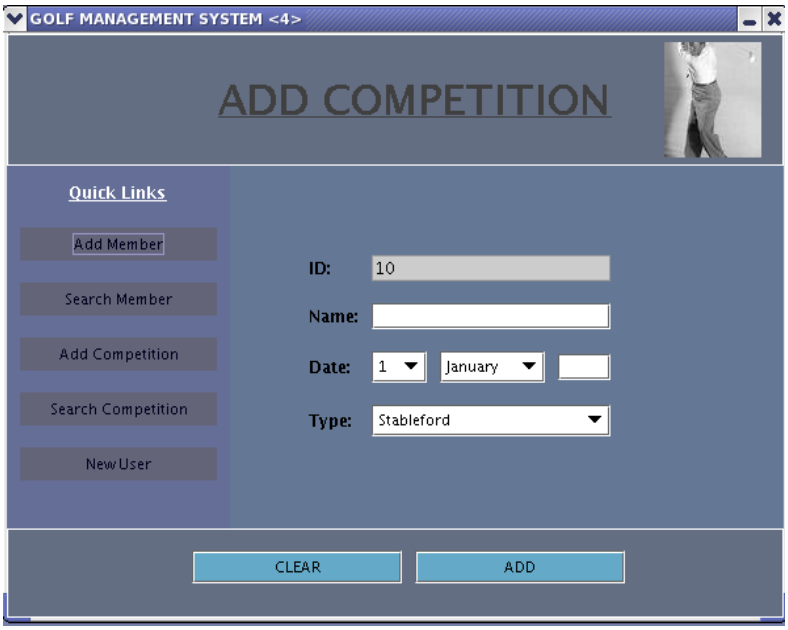



**Adding a competition**

To carry out the task of adding a competitions detail to your database, simply select this button from the main menu. Following this, the interface will appear as shown in figure 1.5. Data input is required

involving date, name and competition type for that competition so the club maintains a record of these. An incrementing ID will automatically be assigned to each competition. Any inputted data can be cleared by selecting the button corresponding to this. Once you are satisfied with your input, select the button to add this competition to your database. As previously stated return to the main menu by selecting the cross.

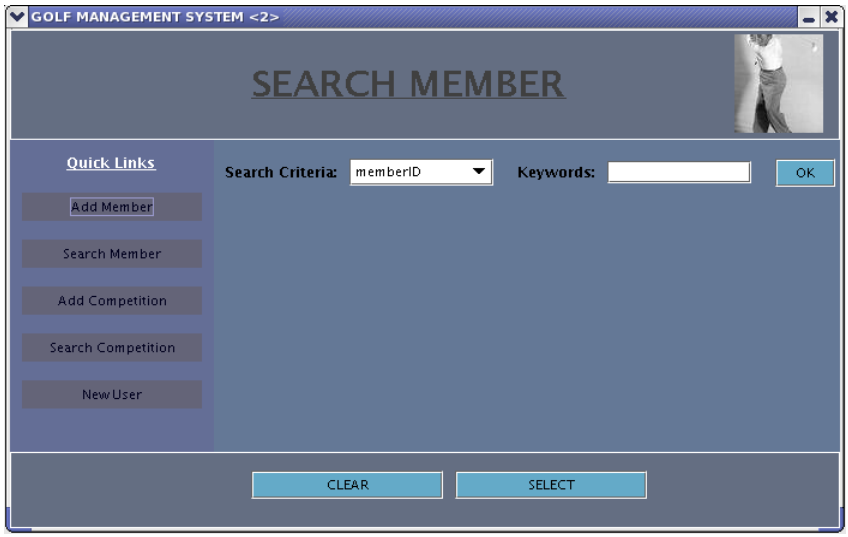
Figure 1.5 Add competition details




**Searching for a member**

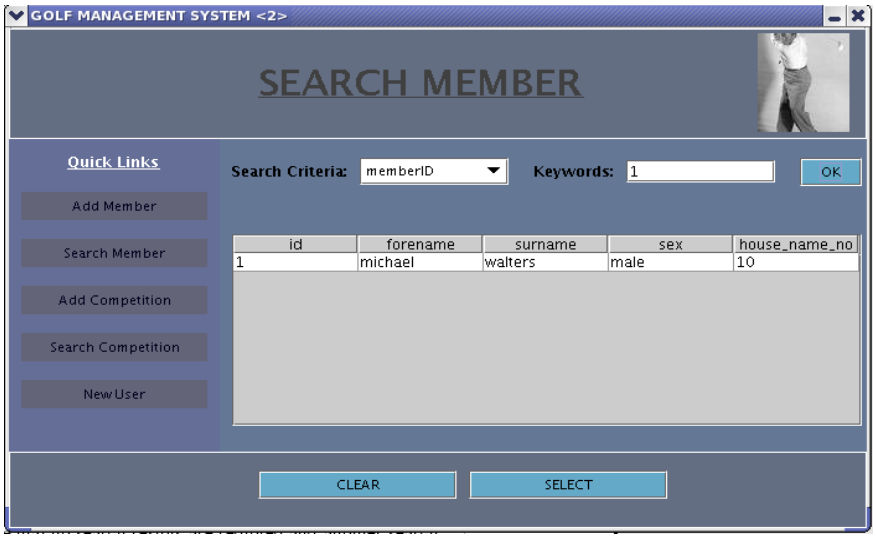
In order to search for a member of your golf club, you must initially select this option from the main menu. An interface will be displayed as shown in figure 1.6. You will notice that certain criteria must be inputted to carry out a search for a member. You are initially required to select a search type from the combo box displayed. Following this, enter a key word for your search and select the OK button.

Figure 1.6 Enter search criteria




As shown in figure 1.7, you must then select your required option from the table of search results. Once one of these records has been highlighted simply select the get record button. If it is the case that no search results are required and another search needs to be conducted, select the clear button to return to the initial state of the interface.

Figure 1.7 Select a result



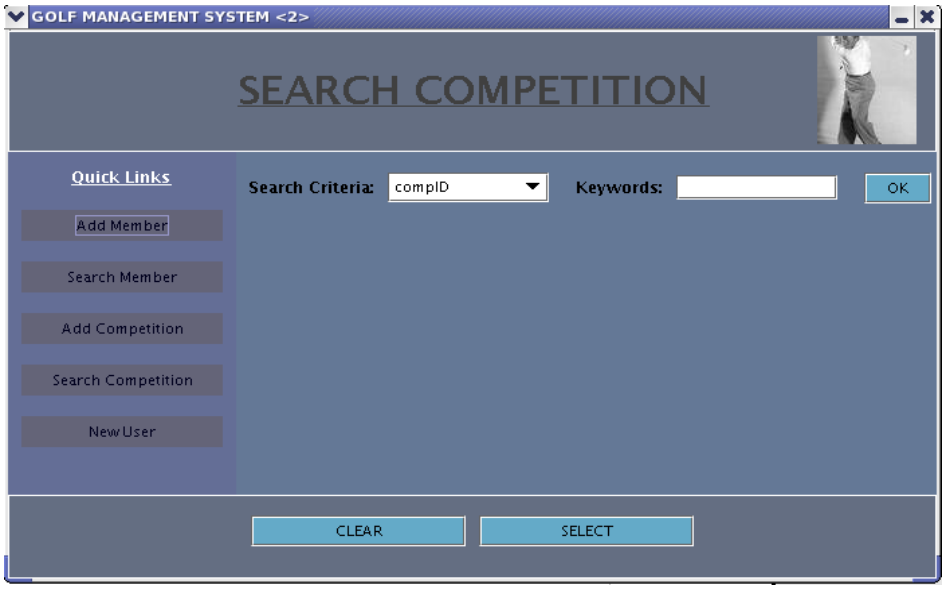
Following this selection an interface will be displayed (figure 1.8) showing that competitions detail. From here you are able to edit these details by selecting the button corresponding to this. Once these details have been modified simply select the update button.

Figure 1.8 Member details

**Searching for a competition**

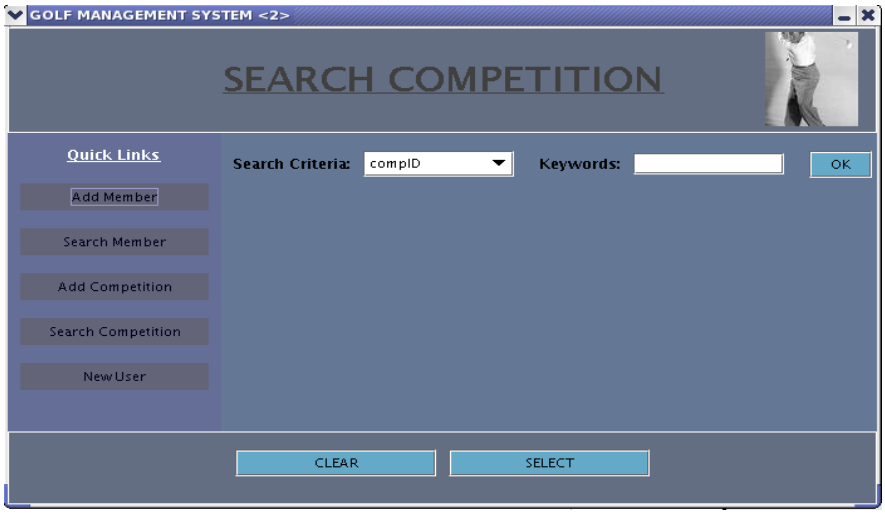
Searching for a competition is very similar to searching for a member. In order to search for a competition of your golf club, you must initially select this option from the main menu. An interface will be displayed as shown in figure 1.9. You will notice that certain criteria must be inputted to carry out a search for a competition. You are initially required to select a search type from the combo box displayed, depending on how you wish to search for a competition. Following this, enter a key word for your search and select the OK button.

Figure 1.9 Enter search criteria



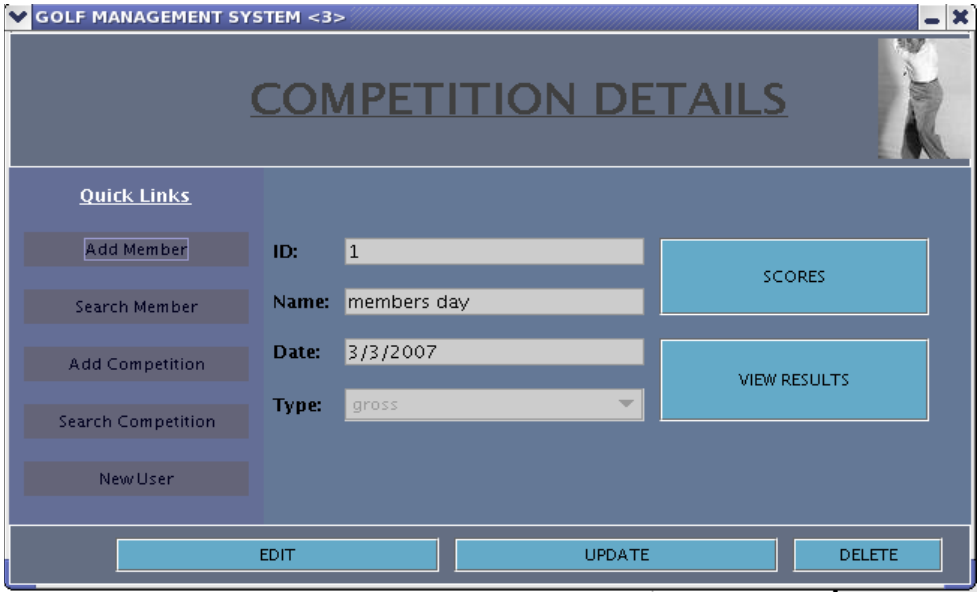

As shown in figure 2.0, you must then select your required option from the table of competition results. Once one of these records has been highlighted simply select the get record button. As previously mentioned, if it is the case that no search results are required and another search needs to be conducted, select the clear button to return to the initial state of the interface.

Figure 2.0 Select a result



Following this selection an interface will be displayed (figure 2.1) showing that competitions detail. From here you are able to edit these details by selecting the button corresponding to this. Once these details have been modified simply select the update button.

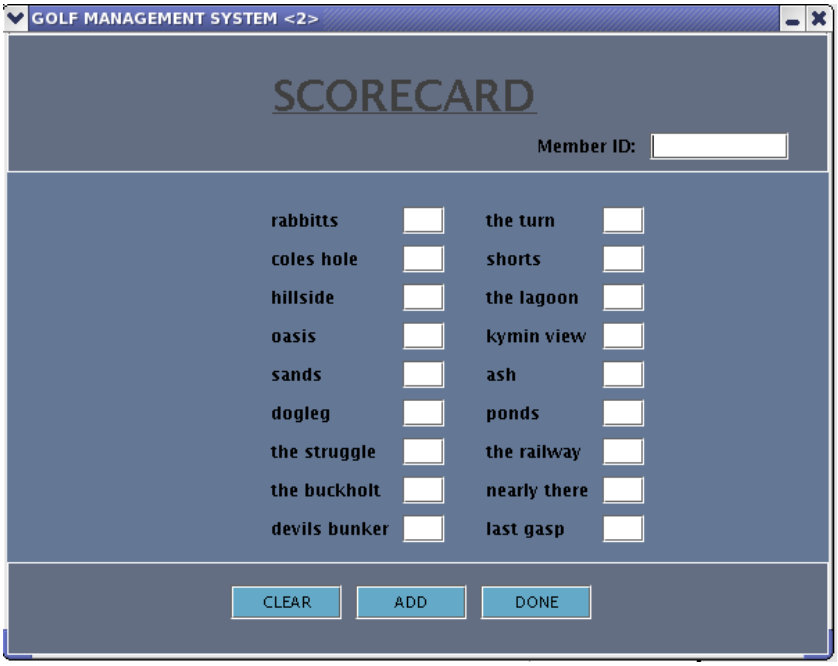
Figure 2.1 Competition details




**Adding scores for a specific competition**

From the competition details screen you can select the add scores button to enter some data from a competition. Figure 2.2 demonstrates this scorecard awaiting input. This is set up so that you are required to enter a members ID along with their competition results. You will notice that you cannot enter more than one set of results for a member in the same competition. Following this if more results are required for input, select the OK button to produce a blank scorecard. Otherwise, select the done button, which returns you to the previous screen.

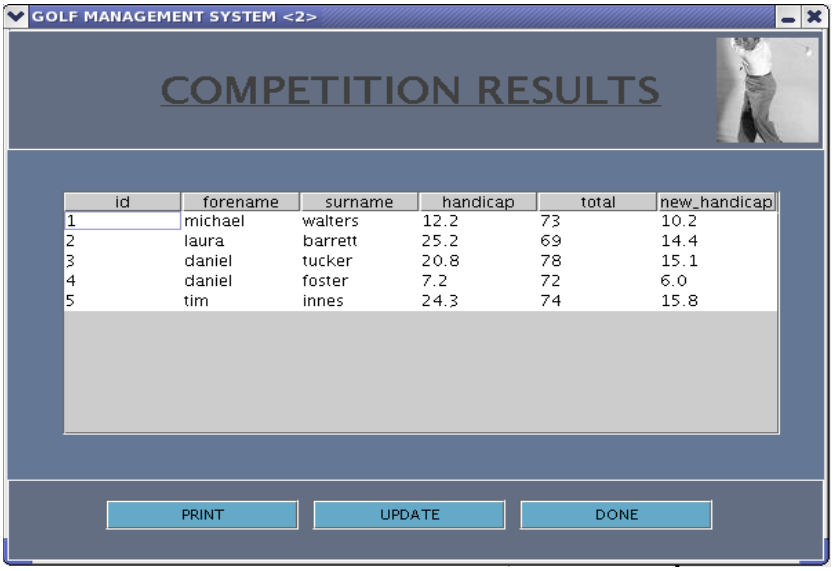
Figure 2.2 Scorecard




**Viewing results for a competition**

Once you've successfully entered all of the results for a competition, select the view results button from the competition details screen. Figure 2.3 shows an example of these competition results. These results take into account the type of the specified competition. They also calculate the new handicap of each member participating in the competition. To update these handicaps for each member select the update button situated below the results table. Following this select the done button to return to the previous screen.

Figure 2.3 View results

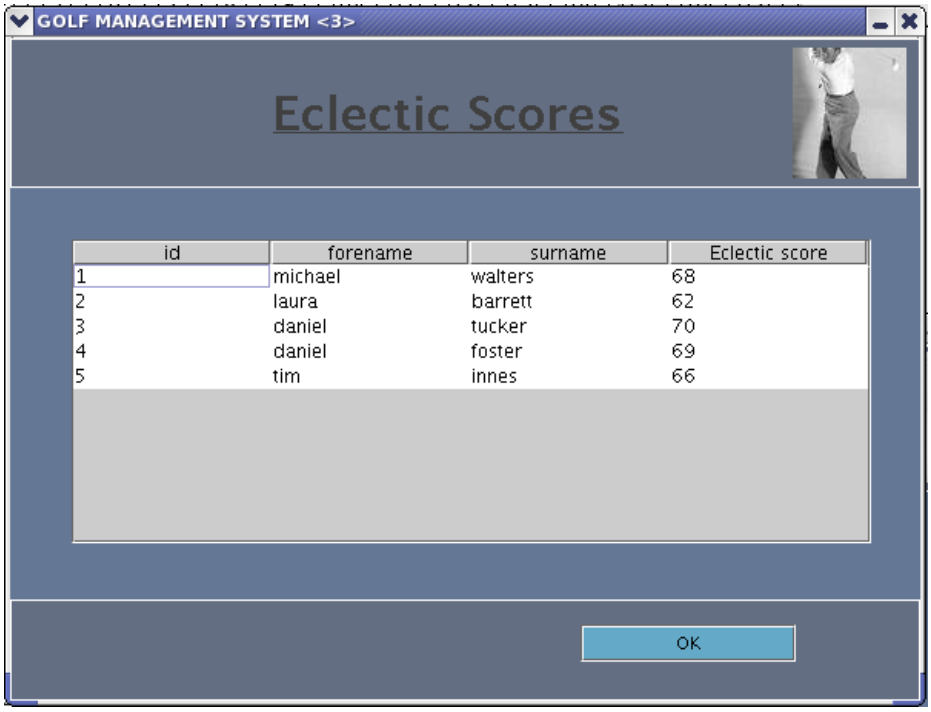



**Finding the top members at your club**

Following the entry of competition results, this system provides a feature, which enables users to discover the members performing the best at their club. This feature can be accessed from the main menu by selecting the corresponding button. Following this, an interface like the one shown in figure 2.4 will be displayed. This splits members up into men's, women's and juniors sections and displays the players improving the most as well as the players who have shot the lowest scores in each of these sections. Furthermore, each member's eclectic score can be found from here by selecting the appropriate button (Figure 2.5). This displays a table of results for all members at the club. You can again return to the main menu by selecting the cross in the corner of the screen.

Figure 2.4 Star members

Figure 2.5 Eclectic scores



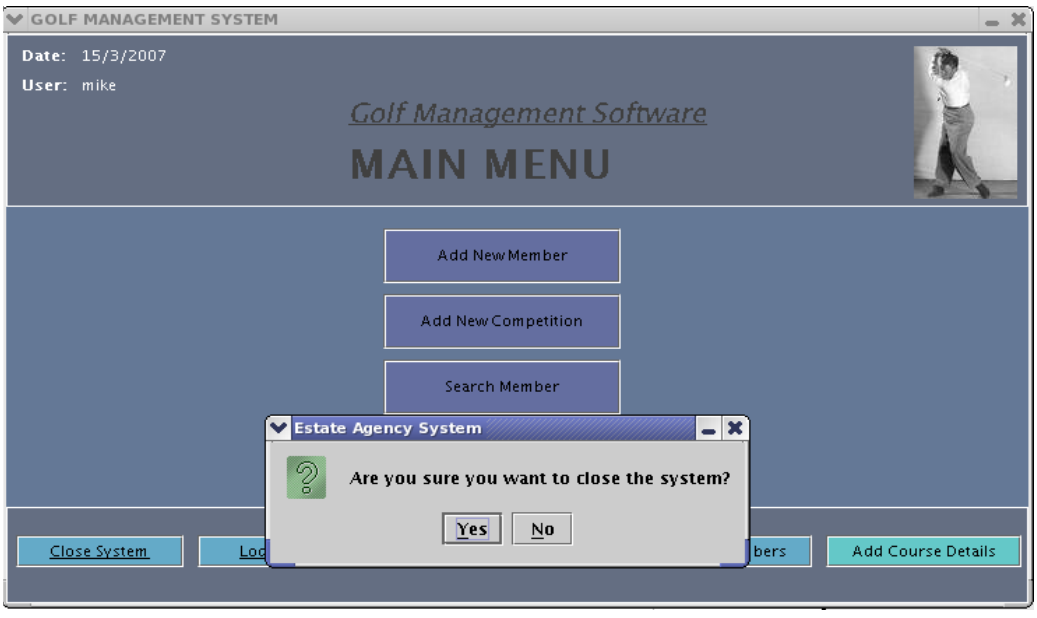
**Navigation help**

The quick links panel provided within many interfaces throughout the application serves to help navigation. It contains all of the main features involved with this golf scoring facility. Therefore an example of this could be when you wish to search for a member whilst still having a competitions detail screen open.

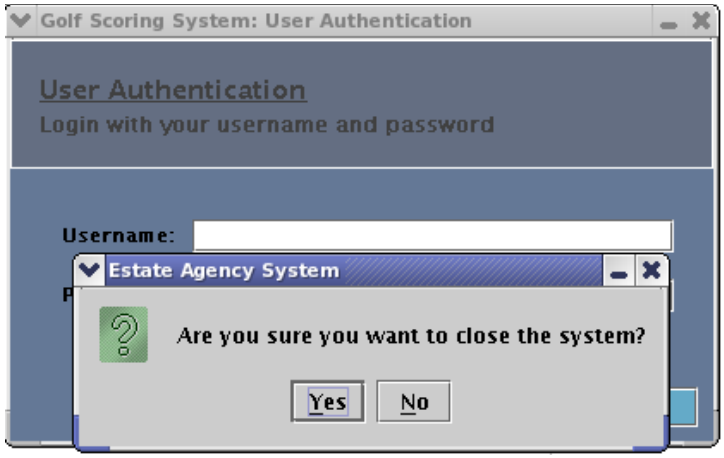
**Closing the system**

The system can be closed from either the main menu or the login interface. You will notice when using this application the main menu will always be running in the background no matter how many additional screens are being used. When closing the system you will be prompted for confirmation as shown below in figure 2.6 from the main menu and in figure 2.7 from the login interface.

2.6 Main menu closing



2.7 Log in closing




**Logging off**

This procedure can be simply carried out from the main menu screen. This will also prompt the user for confirmation as shown below in figure 2.8.

Figure 2.8 Main menu log off